Universidad Complutense de Madrid
Facultad de Informática

Sistemas Informáticos 2011/2012

Constructor: Plataforma como Servicio en la Nube para Startups

Autores:
Adrián Escoms Alonso
Isabel Espinar Pina
Esther Rodrigo Ortiz

Director de proyecto:
José Luis Vázquez-Poletti (Dpto. Arquitectura de Computadores)

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Adrián Escoms Alonso *Isabel Espinar Pina* *Esther Rodrigo Ortiz*

Agradecimientos

El desarrollo de este proyecto no habría sido posible sin todos los apoyos y ayudas que hemos recibido. Empezando por nuestro tutor José Luis Vázquez-Poletti, por su entusiasmo, paciencia y la confianza puesta en nosotros que ha hecho que afrontáramos este desafío con mucha ilusión. También queremos dar las gracias a nuestras familias, sin las cuales no habríamos llegado hasta aquí. Y por último a nuestros amigos, por tantos buenos momentos.

Muchas gracias a todos.

Resumen

A lo largo de este documento se presenta una aplicación que apoyándose en las virtudes de la computación Cloud y gracias a los recursos que Amazon Web Services proporciona, que una empresa de software con escasa posibilidad de inversión, pueda comenzar su actividad y seguir prosperando, disminuyendo los gastos en hardware y en el mantenimiento del mismo.

Se trata de una interfaz gráfica de usuario desde la que se pueden gestionar máquinas virtuales utilizando indirectamente la API de Amazon EC2, configurar dichas máquinas gracias a los recursos de Chef, y todo ello pudiendo llevar la contabilidad de cada una de las fase del proyecto.

Abstract

This paper presents an application relying on the virtues of cloud computing resources and thanks to Amazon Web Services provides, to a software company with a low probability of investment activity, can begin and continue to thrive reducing hardware costs and the maintenance.

This is a graphical user interface from which you can manage virtual machines using indirectly the Amazon EC2 API, set the machinery by Chef resources, and all of thant being able to keep accounts of each project phase.

Índice

1.Computación Cloud

1.1. ¿Qué es la Computación Cloud?

Una de las definiciones más extendidas para definir este concepto es la proporcionada por el National Institute of Standards and Technology - NIST Norteamericano:

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

"Cloud Computing es un modelo que permite acceder a la red, por demanda, un fondo compartido de recursos de cómputo configurables (redes, servidores, almacenamiento, aplicaciones, y servicios) que pueden ser rápidamente implementados y actualizados con un mínimo esfuerzo de administración o interacción de proveedores de servicio."

Hablando de una forma más concreta, Cloud computing es un paradigma de computación emergente donde los datos y servicios residen en centros de datos que pueden ser accedidos desde cualquier dispositivo conectado a internet independientemente de la plataforma usada. Esto permite aumentar el número de servicios basados en la red generando beneficios tanto para los proveedores como para los usuarios.

La mayor ventaja de la Computación Cloud es su simplicidad ya que sus infraestructuras permiten prescindir de la instalación de cualquier tipo de hardware lo que requiere menor inversión para empezar a trabajar y proporciona una mayor capacidad de adaptación.

Figura 1.1: El modelo Cloud. Uno de los objetivos principales del Cloud es ofrecer servicios informáticos de cualquier índole independientemente del dispositivo que estemos empleando o dónde se encuentren los recursos a los que queremos acceder.

1.2. Caracteristicas

Algunas de las características mas importantes de la Computación Cloud son las que se describen a continuación:

- **Auto Reparable**: En caso de fallo, el último backup de la aplicación pasa a ser automáticamente la copia primaria y se genera uno nuevo.

- **Virtualizado**: Las aplicaciones son independientes del hardware en el que las ejecutemos. Varias aplicaciones pueden ser ejecutadas en una misma máquina o una aplicación puede usar varias máquinas a la vez.

- **Seguridad**: La distribución de los datos en numerosos servidores y la capacidad de desviar recursos propios de esta tecnología aumenta la seguridad en comparación con los centros de datos tradicionales. El sistema está creado de tal forma que permite a diferentes clientes compartir la infraestructura sin preocuparse de ello y sin comprometer su seguridad y privacidad.

- **Accesibilidad.** Gracias a las nuevas tecnologías, las aplicaciones en cloud están “libres” en la red y disponibles para los usuarios, que podrán acceder a ellas mediante PC, portátiles o incluso desde teléfonos móviles.

- **Aplicaciones “a la carta”.** El usuario puede, en todo momento, decidir qué aplicaciones usar y elegir entre aquellas que son gratuitas y las que no lo son. En el caso de las aplicaciones de pago, el coste irá en función de diversas variables, como el servicio contratado, el tiempo que se ha usado ese servicio, el volumen de tráfico de datos utilizado, etc.

- **Escalabilidad y elasticidad “self-service**”: Las aplicaciones en cloud son totalmente elásticas en cuanto a su rapidez de implementación y adaptabilidad. Además, son totalmente escalables, es decir, hoy podemos estar utilizando sólo

un 10% del total de la aplicación y mañana podemos acceder al 80% de la misma con total normalidad y rapidez.

- **Supervisión del servicio.** Los sistemas en cloud controlan y optimizan el uso de los recursos de manera automática, por lo que el uso de estos puede seguirse, controlarse y notificarse, lo que aporta transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

1.3. Evolución de la Computación Cloud

La computación en nube ha recorrido un largo camino desde que fue marcada por primera vez como una perspectiva de futuro por parte de algunos investigadores. La historia inicial de la computación en nube nos lleva a finales del siglo veinte, cuando la prestación de servicios de computación comenzó. Sin embargo el concepto se remonta a J.C.R. Licklider y John McCarthy.

En 1996, Douglas Parkhill con su libro llamado “El desafío de la utilidad de la computadora” exploró a fondo muchas de las características actuales de la computación en nube (aprovisionamiento elástico a través de un servicio de utilidad), así como la comparación de la industria eléctrica y el uso de las formas públicas, privadas, comunitarias y gubernamentales. Pero otros investigadores afirman que las raíces de la computación en nube nos llevan hasta la década de 1950 con las observaciones de Herb Grosch. Él decía que la potencia de una computadora es proporcional al cuadrado de su precio (Ley Grosch), sin embargo la ley de Moore se encargó de desmentir esto. Algunos académicos recientemente han rehabilitado la ley de Grosch, mirando la historia de la computación en la nube, afirman que “Grosch estaba equivocado sobre el modelo del costo de la computación en nube, no se equivocaba en su suposición de que las economías eficientes y adaptables podría alcanzar su objetivo si confían en centros de datos centralizados en lugar confiar en el almacenamiento de unidades“.

Las empresas de telecomunicaciones hasta la década de los 90s eran quienes ofrecían redes privadas virtuales (VPN) con una calidad de servicio semejante, pero a un costo mucho menor. Al ser capaces de equilibrar el tráfico pudieron hacer uso del ancho de banda total de la red con mayor eficacia. Incluso el símbolo de la nube se utiliza para indicar el punto de demarcación entre lo que es la responsabilidad del proveedor y lo que era la responsabilidad del usuario. Ahora la computación en nube extiende este límite para cubrir servidores, así como la infraestructura de red.

Uno de los pioneros en la computación en nube fue Salesforce.com, que en 1999 introdujo el concepto de entrega de aplicaciones empresariales a través de una sencilla página web. Amazon era el siguiente en el tren, al lanzar Amazon Web Service en 2002. Entonces llegó Google Docs en 2006, que realmente trajo el cloud computing a la vanguardia de la conciencia del público. 2006 también vio la introducción de Elastic Compute Cloud de Amazon (EC2) como un servicio web comercial que permitió a las empresas pequeñas y particulares alquilar equipos en los que pudieran ejecutar sus propias aplicaciones informáticas.

Esto fue seguido por una colaboración de toda la industria en 2007 entre Google, IBM y una serie de universidades de los Estados Unidos. Luego vino Eucalyptus en 2008, como la primera plataforma de código abierto compatible con el API-AWS para el despliegue de clouds privados, seguido por OpenNebula, el primer software de código abierto para la implementación de nubes privadas e híbridas. Microsoft entraría hasta el 2009 con el lanzamiento de Windows Azure. Luego en 2010 proliferaron servicios en distintas capas de servicio: Cliente, Aplicación, Plataforma, Infraestructura y Servidor. En 2011, Apple lanzó su servicio iCloud, un sistema de almacenamiento en la nube, para documentos, música, vídeos, fotografías, aplicaciones y calendarios que prometía cambiar la forma en que usamos la computadora.

1.4. Tipos de nube

Para atender a las diferentes necesidades de las empresas existen varios tipos de nubes, dependiendo de donde se encuentren instaladas las aplicaciones y qué clientes pueden usarlas tendremos nubes públicas, privadas o híbridas, cada una de ellas con sus ventajas e inconvenientes.

1.4.1. Nubes públicas

Para atender a las diferentes necesidades de las empresas existen varios tipos de nubes, dependiendo de donde se encuentren instaladas las aplicaciones y qué clientes pueden usarlas tendremos nubes públicas, privadas o híbridas, cada una de ellas con sus ventajas e inconvenientes.

En las nubes públicas, los servicios que se ofrecen se encuentra en servidores externos al usuario, pudiendo tener acceso a las aplicaciones de forma gratuita o de pago.

La ventaja más clara de las nubes públicas es la capacidad de procesamiento y almacenamiento sin instalar máquinas localmente, por lo que no tiene una inversión inicial o gasto de mantenimiento en este sentido, si no que se

paga por el uso. La carga operacional y la seguridad de los datos (backup, accesibilidad, etc.) recae íntegramente sobre el proveedor del hardware y software, debido a ello, el riesgo por la adopción de una nueva tecnología es bastante bajo. El retorno de la inversión se hace rápido y más predecible con este tipo de nubes.

Como inconvenientes se cuenta con el acceso de toda la información a terceras empresas, y la dependencia de los servicios en línea. También puede

resultar difícil integrar estos servicios con otros sistemas propietarios. Es muy importante a la hora de apostar por un servicio en la nube pública, asegurarse de que se puede conseguir todos los datos que se tengan en ella gratuitamente y en el menor tiempo posible.

Figura 1.2: Tipos de nube, características y modelos de servicio del Cloud

1.4.2. Nubes privadas

En las nubes privadas, sin embargo, la plataforma se encuentra dentro de las instalaciones del usuario de la misma y no suele ofrecer servicios a terceros. En general, una nube privada es una plataforma para la obtención solamente de hardware, es decir, máquinas, almacenamiento e infraestructura de red (IaaS), pero también se puede tener una nube privada que permita desplegar aplicaciones (PaaS) e incluso aplicaciones (SaaS).

Como ventaja de este tipo de nubes, al contrario que las públicas, es la localización de los datos dentro de la propia empresa, lo que conlleva a una mayor seguridad de estos, corriendo a cargo del sistema de información que se utilice. Incluso será más fácil integrar estos servicios con otros sistemas propietarios.

Sin embargo, como inconveniente se encuentra la inversión inicial en infraestructura física, sistemas de virtualización, ancho de banda y seguridad, lo que llevará a su vez a pérdida de escalabilidad y desescabilidad de las plataformas, sin olvidar el gasto de mantenimiento que requiere. Esta alta inversión supondrá un retorno más lento de la inversión.

1.4.3. Nubes hibridas

Las nubes híbridas consiste en combinar las aplicaciones locales con las de la nube pública. Se puede ver también como aplicación privada que se ve aumentada con los servicios de Cloud Computing y la infraestructura. Esto

permite a una empresa mantener el control de sus principales aplicaciones, al tiempo de aprovechar el Cloud Computing en los lugares donde tenga sentido.

Por ejemplo, muchas empresas han visto que es más económico usar un IaaS, como por ejemplo Amazon Simple Storage Service (S3), para almacenar imágenes, vídeos y documentos que en infraestructuras propias. El modelo híbrido también se presta a un enfoque incremental.

Incluso la nube híbrida puede ser un buen paso intermedio antes de pasar la mayor parte de las aplicaciones a la nube, ya que es algo menos arriesgado. Por tanto, sería interesante pasar algunas aplicaciones más útiles para la nube a esta y en el momento que se esté más cómodo, mover las que sean necesarias.

Una nube híbrida tiene la ventaja de una inversión inicial más moderada y a la vez contar con SaaS, PaaS o IaaS bajo demanda. En el momento necesario, utilizando las APIs de las distintas plataformas públicas existentes, se tiene la posibilidad de escalar la plataforma todo lo que se quiera sin invertir en infraestructura con la idea de tomar uno de los siguientes caminos:

- Si dicha necesidad llegara a ser de carácter estable, sería recomendable incrementar la capacidad de la nube privada e incorporar los servicios adoptados en la pública pasandolos a la nube propia.

- Si dicha necesidad es puntual o intermitente se mantendría el servicio en los Clouds públicos, lo que permite no aumentar la infraestructura innecesariamente.

Parece que este tipo de nubes está teniendo buena aceptación en las empresas de cara a un futuro próximo, ya que se están desarrollando softwares de gestión de nubes para poder gestionar la nube privada y a su vez adquirir recursos en los grandes proveedores públicos.

1.5. Modelos de servicios en Cloud

Cloud Software As a Service (SaaS): Saas es aquella aplicación ofrecida por un fabricante de software o proveedor de servicios informáticos a través de internet, para su uso o utilización por varios clientes. El fabricante es el que en última instancia se ocupa del manteniendo de la privacidad de los datos y la personalización de la aplicación.

En este modelo de servicio, el usuario paga por el uso y por la infraestructura necesaria (almacenamiento, seguridad, alojamiento, etc.) para el correcto funcionamiento de la aplicación y, a excepción de unos pocos parámetros de configuración, se limita a utilizar la herramienta y sus funcionalidades.

Cloud Platform As a Service (PaaS): Este modelo de nube amplía las prestaciones del caso anterior, de forma que el consumidor o usuario de esa nube, puede desplegar en ella aplicaciones desarrolladas o adquiridas por él mismo, en pos de ampliar las funcionalidades de dicha nube. Todo esto, por supuesto, se deberá desarrollar en aquellos lenguajes de programación que sean aceptados por el proveedor de la nube.

En este modelo de nube, el usuario no podrá gestionar la infraestructura de la nube, pero tendrá acceso tanto sobre las aplicaciones desplegadas en ella como sobre la configuración de las diversas herramientas que utilice.

Cloud Infrastructure As a Service (IaaS): En el IaaS, se parte de la idea de la externalización de servidores para espacio en disco, base de datos etc., en lugar de tener un control completo de los mismos con el DATA CENTER dentro de la empresa, u optar por un centro de datos y sólo administrarlo. Mediante este modelo de despliegue en Cloud, lo que se tiene es una solución basada en la virtualización, en la que se paga por el nivel de consumo de los recursos: espacio en disco utilizado, tiempo de CPU, espacio en base de datos, transferencia de datos.

La ventaja más inmediata de elegir este tipo de soluciones es la de desplazar una serie de problemas al proveedor relacionados con la gestión de las máquinas y llegar a un ahorro de costes importante, ya que pagaremos solo por lo consumido en función del nivel servicio que nos ofrezca dicho proveedor.

Otro aspecto fundamental a tener en cuenta, es que las Infraestructura como servicio pueden permitir una escalabilidad automática o semiautomática, de forma que podamos contratar más recursos según los vayamos necesitando.

1.6. Cloud Computing en cifras

Cloud Computing tiene millones de usuarios, muchos de los cuales seguramente no saben que lo están usando. A continuación expondremos algunas cifras de usuarios y servicios que utilizan esta tecnología.

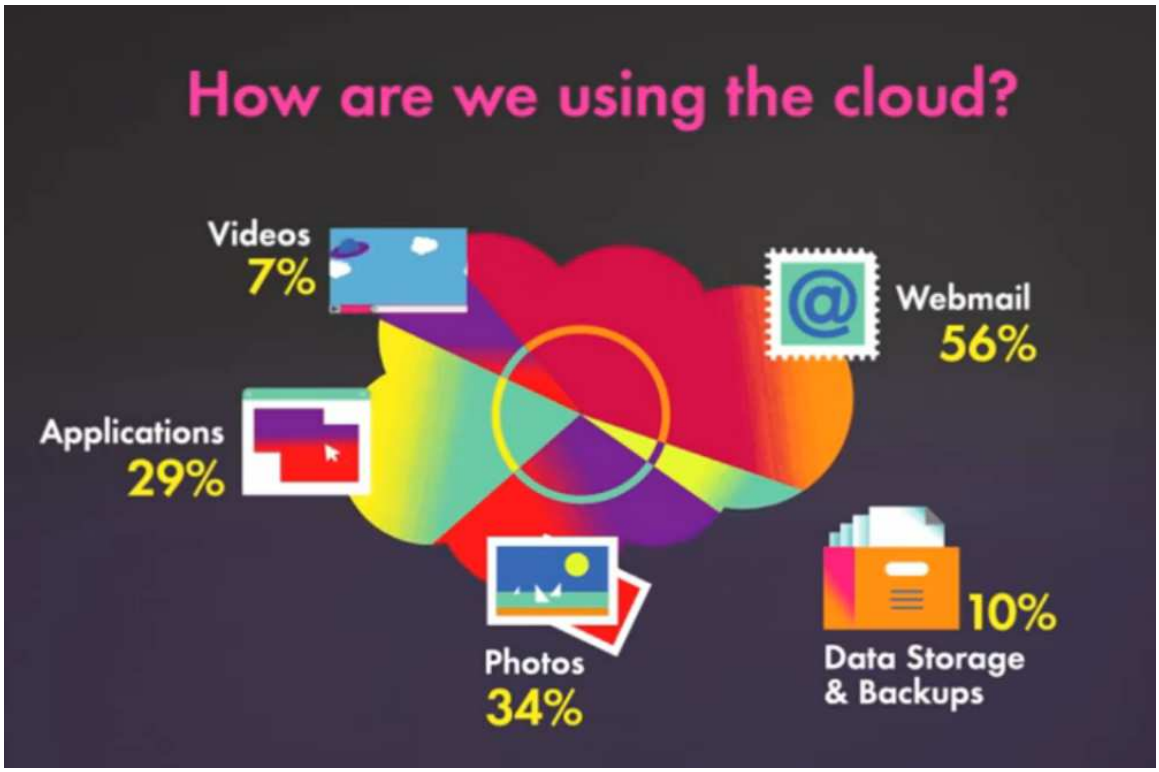


Figura 1.3: Uso del Cloud

Los servicios más usados de la computación Cloud, son los destinados al envío y recepción de mensajes. El 40% de estas operaciones se realizan desde dispositivos móviles. Los principales suministradores de este servicio son Yahoo!, Msn y Gmail con 106, 47 y 37 millones de usuarios mensuales correspondientemente.

Las redes sociales y aplicaciones que permiten la comunicación sin importar las distancias acaparan otro gran porcentaje de los servicios más usados de Cloud Computing.

Skype tiene 443 millones de suscriptores los cuales hablaron más de 23.6 billones de minutos el año 2009. En cuanto a las redes sociales, los usuarios de twitter envían 90 millones de tweets al día, y Facebook es la que más seguidores tiene, superando recientemente los 900 millones de usuarios, y con una previsión de llegar a 1000 millones en este 2012.

Cloud Computing también se utiliza para aplicaciones que permiten el almacenamiento de archivos, como por ejemplo Dropbox, que tiene unos 25 millones de usuarios, los cuales almacenan en sus servidores más de 200 millones de archivos cada día.

En 2010 los servicios de la nube generaron 68.3 billones de dólares de ingresos y se prevé que para 2014 el beneficio asciende a 148.8 billones de dólares.

A continuación se puede observar algunas de las empresas que usan esta tecnología para ofrecer sus servicios y la forma en la que usan la nube para ofrecerlos.

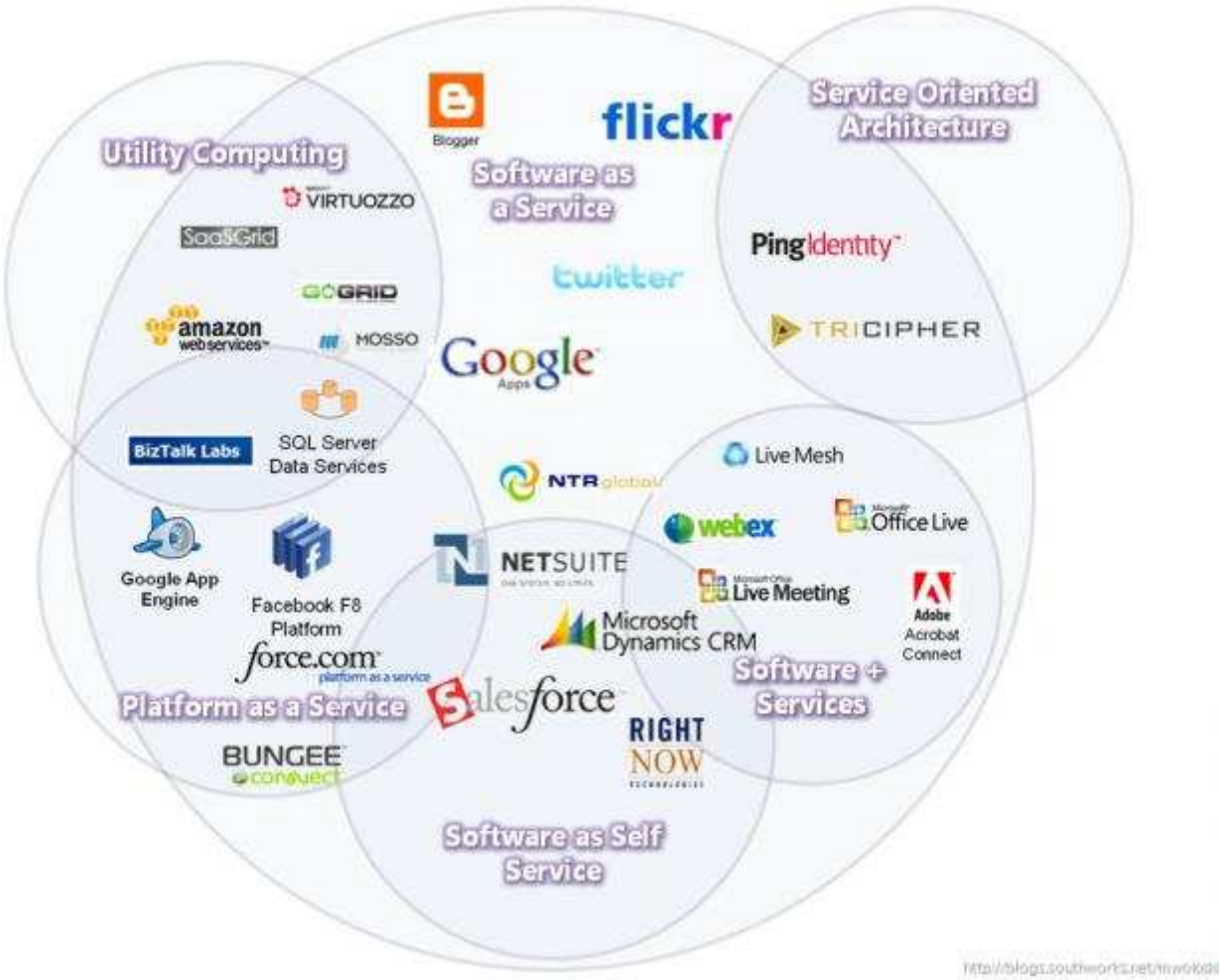


Figura 1.4: Empresas que usan Cloud computing

2.Amazon Web Service

2.1. ¿Qué es la AWS?

Amazon Web Services (AWS) ofrece a empresas de todos los tamaños una plataforma de servicios web de infraestructura basada en la nube. Con AWS puede solicitar potencia informática y capacidad de almacenamiento, así como otro tipo de servicios que permiten obtener acceso a un conjunto de servicios de infraestructura de TI elásticos, tal y como la empresa los necesita. AWS ofrece flexibilidad para poder elegir la plataforma de desarrollo o el modelo de programación que mejor se adapte a los problemas que se estén intentando resolver. Hay que pagar únicamente por lo que se usa, sin ningún tipo de gastos por adelantado ni compromisos a largo plazo.

De este modo, AWS se convierte en la forma más rentable de ofrecer una aplicación a los clientes. Además, con AWS, se puede utilizar la infraestructura de computación internacional de Amazon.com, columna vertebral de la empresa

transaccional valorada en varios miles de millones de dólares y cuya infraestructura informática distribuida escalable, fiable y segura se ha perfeccionado durante más de diez años.

Amazon Web Services ofrece, tanto a desarrolladores como a organizaciones de TI numerosas ventajas, entre las que se incluyen:

- Rentabilidad. Se paga únicamente por el consumo realizado, a medida que se utilice y sin ningún tipo de compromiso por adelantado. A medida que la nube de Amazon Web Services crezca, sus costes de explotación, gestión y hardware se irán reduciendo. escalabilidad.
- Fiabilidad. Ofrece al cliente la posiblidad de utilizar una infraestructura web probada en complicadas situaciones capaz de gestionar lo que necesite. La nube de Amazon Web Services destaca por su seguridad, fiabilidad y distribución, ofreciendo elevados niveles de fiabilidad y enormes posibilidades de escalabilidad.
- Flexibilidad. Puede crear la aplicación que desee con cualquier plataforma o modelo de programación. El cliente será quien controle los recursos que consumirá y los adaptará a su aplicación según sea necesario.
- Global. No es necesario empezar desde cero. Amazon Web Services ofrece varios servicios que puede incorporar a las aplicaciones. Desde bases de datos hasta pagos, estos servicios ayudan a crear fantásticas aplicaciones de una forma rentable y con la menor inversión por adelantado.

2.1.1. Una plataforma flexible para su negocio

Amazon Web Services no obliga a utilizar ningún tipo de sistema operativo, plataforma de desarrollo ni modelo de programación concreto. Se adapa a la necesidades del negocio, pudiendo la empresa tomar la decisión más adecuada. Ayudando a la empresa empezar a ahorrar dinero, tiempo, dolores de cabeza y molestias.

- Independencia de plataformas – El usuario puede elegir el sistema operativo, el modelo de programación, la configuración.
- Listo desde el primer día – AWS puede ejecutarse en cualquiera de los sistemas que tenga en su centro de datos.
- Servicios de aplicación – AWS ofrece atractivos servicios para la gestión de bases de datos (Amazon SimpleDB), colas (Amazon SQS), pagos (Amazon FPS) y mucho más.

2.1.2. Kit de herramientas AWS para Eclipse

El Kit de herramientas AWS para Eclipse es un complemento de código abierto para Eclipse Java IDE que facilita a los desarrolladores las tareas de desarrollo, depuración e implementación de aplicaciones Java mediante Amazon Web Services. Con el Kit de herramientas AWS para Eclipse pudiendo empezar a

trabajar más rápido, aumentando sus niveles de productividad a la hora de crear aplicaciones AWS. El Kit de herramientas AWS para Eclipse incluye:

- ***SDK AWS para Java:*** El Kit de herramientas AWS para Eclipse incluye cómodamente el SDK AWS para Java, para que el cliente pueda empezar a crear aplicaciones Java en los servicios de infraestructura AWS en Eclipse, incluyendo Amazon S3, Amazon EC2 y Amazon SimpleDB.
- ***Gestión de Amazon SimpleDB:*** Permite administrar los datos de Amazon SimpleDB sin escribir una sola línea de código. Basado en el proyecto Eclipse Data Tools Platform, el Kit de herramientas AWS para Eclipse ofrece una interfaz gráfica que le facilita la tarea de gestión de sus dominios, elementos y atributos de Amazon SimpleDB.
- ***Gestión de Amazon EC2:*** Basado en Eclipse Web Tools Platform, el Kit de herramientas AWS para Eclipse guía a los desarrolladores de Java mediante flujos de trabajo comunes y automatiza la configuración de herramientas, con tareas tales como la configuración de conexiones a depurador remoto y la gestión de contenedores Tomcat. Los pasos necesarios para configurar servidores Tomcat, ejecutar aplicaciones en Amazon EC2 y depurar el software de forma remota ahora se realizan de forma transparente a través de Eclipse IDE.

2.1.3. Requisitos previos

- Requiere Java 1.5 o superior.
- Requiere Eclipse IDE for Java Developers 3.5 o superior. Eclipse IDE for Java EE Developers 3.6 recomendado.
- Para Gestión de Amazon EC2:
  - Eclipse Web Tools Platform 2.0 o superior, en función de los requisitos de su distribución Eclipse. Eclipse IDE for Java EE Developers tiene preinstalada la Web Tools Platform.
  - Debe estar inscrito en Amazon EC2.

- Para Gestión de Amazon SimpleDB:
  - Eclipse Data Tools Platform 1.7 o superior. Eclipse IDE for Java EE Developers tiene preinstalada la Data Tools Platform.
  - Debe estar inscrito en Amazon SimpleDB.

2.2. Amazon Elastic Compute Cloud EC2

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores recursos informáticos escalables y basados en web.

La sencilla interfaz de servicios web de Amazon EC2 permite obtener y configurar su capacidad con una fricción mínima. Proporciona un control completo sobre sus recursos informáticos y permite ejecutarse en el entorno informático acreditado de Amazon. Amazon EC2 reduce el tiempo necesario para obtener y arrancar nuevas instancias de servidor en minutos, lo que permite escalar rápidamente la capacidad, ya sea aumentándola o reduciéndola, según cambien sus necesidades. Amazon EC2 cambia el modelo económico de la informática, al permitir pagar sólo por la capacidad que utiliza realmente. Amazon EC2 proporciona a los desarrolladores las herramientas necesarias para crear aplicaciones resistentes a errores y para aislarse de los casos de error más comunes.

2.2.1. Funcionalidad de Amazon EC2

Amazon EC2 presenta un auténtico entorno informático virtual, que permite utilizar interfaces de servicio web para iniciar instancias con distintos sistemas operativos, cargarlas con su entorno de aplicaciones personalizadas, gestionar sus permisos de acceso a la red y ejecutar su imagen utilizando los sistemas que desee. Para utilizar Amazon EC2, el usuario necesita:

- Seleccionar una imagen de plantilla preconfigurada para empezar a funcionar inmediatamente. O bien crear una AMI (Amazon Machine Image) que contenga sus aplicaciones, bibliotecas, datos y valores de configuración asociados.

- Configurar la seguridad y el acceso a red en su instancia de Amazon EC2.

- Seleccionar los tipos de instancias y sistemas operativos que desee y, a continuación, iniciar, finalizar y supervisar tantas instancias de su AMI como sea necesario, a través de las API de servicio web o la variedad de herramientas de gestión proporcionadas.

- Determinar si desea una ejecución en varias localizaciones, utilizar puntos finales de IP estáticos o adjuntar almacenamiento de bloques continuo a sus instancias.

- Pagar sólo por los recursos que realmente consuma, como las horas de uso de instancias o la transferencia de datos.

2.2.2. Aspectos destacados del servicio

- **Elastic:** Amazon EC2 permite aumentar o reducir la capacidad en cuestión de minutos, sin esperar horas ni días. Puede enviar una, cientos o incluso miles de instancias del servidor simultáneamente.

- **Totalmente controlado:** El usuario tendrá control total sobre sus instancias. Con acceso de usuario raíz a todas ellas, e interactuando con ellas como con cualquier otra máquina. Puede detener su instancia y mantener los datos en su partición de arranque, para reiniciar a continuación la misma instancia a través de las API del servicio web. Las instancias se pueden reiniciar de forma remota mediante las API del servicio web. Asimismo, tiene acceso a la emisión de consola de sus instancias.

- **Flexible**: Tendrá también la posibilidad de elegir entre varios tipos de instancia, sistemas operativos y paquetes de software. Amazon EC2 permite seleccionar una configuración de memoria, CPU, almacenamiento de instancias y el tamaño de la partición de arranque óptimo para su sistema operativo y su aplicación. Por ejemplo, entre sus opciones de sistemas operativos se incluyen varias distribuciones de Linux y Microsoft Windows Server.

- Con un **diseño** pensado para su uso con otros Amazon Web Services – Amazon EC2 trabaja con Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon SimpleDB y Amazon Simple Queue Service (Amazon SQS) para proporcionar una solución informática completa, procesamiento de consultas y almacenamiento en una gran gama de aplicaciones.

- **Fiable:** Amazon EC2 ofrece un entorno muy fiable en el que las instancias de sustitución se pueden enviar con rapidez y anticipación. El servicio se ejecuta en los centros de datos y la infraestructura de red acreditados de Amazon. El compromiso del contrato a nivel de servicio de Amazon EC2 es de una disponibilidad del 99,95% en cada Región de Amazon EC2.

- **Seguro**: Amazon EC2 ofrece diversos mecanismos para proteger los recursos informáticos.

- **Económico:** Amazon EC2 permite disfrutar de las ventajas financieras de Amazon. El usuario pagará una tarifa muy baja por la capacidad informática que realmente utiliza.

- **On-Demand Instances** – Con On-Demand Instances se puede pagar por la capacidad informática por hora, sin compromisos a largo plazo. Esto liberará al usuario de los costes y las complejidades de la planificación, la compra y el mantenimiento del hardware y transformará lo que normalmente son grandes costes fijos en costes variables mucho más pequeños. Gracias a On-Demand Instances también se elimina la necesidad de comprar una "red de seguridad" de capacidad para gestionar picos de tráfico periódicos.
- **Instancias reservadas** – Las instancias reservadas ofrecen la opción de realizar un pago puntual reducido por cada instancia que desee reservar y recibir a cambio un descuento importante en el cargo de uso por horas de dicha instancia. Existen tres tipos de instancias reservadas (instancias reservadas de utilización ligera, media e intensa) que permiten equilibrar el importe del pago anticipado a realizar con su precio por hora efectivo.
- **Spot Instances** – Con Spot Instances, los clientes pueden ofertar la capacidad sin utilizar de Amazon EC2 y ejecutar dichas instancias mientras su oferta supere el precio actual al contado. El precio puntual cambia periódicamente según la oferta y la demanda, y los clientes cuyas ofertas alcancen o excedan dicho precio tendrán acceso a las instancias puntuales disponibles. Si es flexible respecto a cuándo ejecutar sus instancias, Spot Instances puede reducir significativamente sus costes de Amazon EC2.

2.2.3. Características

Amazon EC2 incluye una serie de potentes funciones para construir aplicaciones escalables, resistentes a fallos y de clase empresarial como las siguientes:

- **Amazon Elastic Block Store** – Amazon Elastic Block Store (EBS) ofrece almacenamiento persistente para instancias de Amazon EC2. Los volúmenes Amazon EBS ofrecen un almacenamiento fuera de la instancia que persiste con independencia de la vida de una instancia. Los volúmenes de Amazon EBS son

volúmenes muy fiables y con una gran disponibilidad, que se pueden utilizar como particiones de arranque de la instancia de Amazon EC2 o bien conectarse a una instancia de Amazon EC2 en ejecución como dispositivo de bloques estándar. Si se utiliza como partición de arranque, las instancias de Amazon EC2 se pueden detener y reiniciar, lo que permite pagar solo por los recursos

de almacenamiento que se utilizan, mientras mantiene el estado de su instancia. Los volúmenes de Amazon EBS ofrecen una duración muy mejorada

con respecto a los almacenes de instancias locales de Amazon EC2, ya que los volúmenes de Amazon EBS se replican automáticamente en segundo plano (en una única Zona de disponibilidad). Para aquellos usuarios que deseen contar con una mayor duración, Amazon EBS proporciona la capacidad de crear instantáneas puntuales coherentes de sus volúmenes, que se almacenarán en Amazon S3 y se replicarán automáticamente en distintas Zonas de disponibilidad. Estas instantáneas se pueden utilizar como punto de partida para nuevos volúmenes de Amazon EBS, y pueden proteger los datos para lograr su duración a lo largo del tiempo.

- **Varias ubicaciones** – Amazon EC2 ofrece la posibilidad de colocar instancias en distintas ubicaciones. Las ubicaciones de Amazon EC2 se componen de Regiones y Zonas de disponibilidad. Las Zonas de disponibilidad son regiones diferentes que están diseñadas para estar aisladas de fallos que se produzcan en otras Zonas de disponibilidad, y que proporcionan conectividad de red de baja latencia a otras Zonas de disponibilidad de la misma Región. Al iniciar instancias en Zonas de disponibilidad distintas, el usuario puede proteger a sus aplicaciones en caso de error de una única ubicación. Las Regiones están compuestas por una o más Zonas de disponibilidad, están geográficamente dispersas y se encuentran en áreas geográficas o países diferentes. El compromiso del contrato de nivel de servicio de Amazon EC2 es de una disponibilidad del 99,95% en cada región de Amazon EC2. Amazon EC2 está disponible actualmente en ocho regiones: EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), EE.UU. Oeste (Norte de California), UE (Irlanda), Asia-

Pacífico (Singapur), Asia-Pacífico (Tokio), América del Sur (São Paulo) y AWS GovCloud.

- **Direcciones Elastic IP** – Las direcciones Elastic IP son direcciones IP estáticas diseñadas para la informática dinámica en nube. Una dirección Elastic IP está asociada a su cuenta, no a una instancia concreta, y puede controlar esta dirección hasta que decida, explícitamente, liberarla. Al contrario que las tradicionales direcciones IP estáticas, las direcciones de Elastic IP permiten disimular los errores en instancias o Zonas de disponibilidad, al reasignar de forma programada sus direcciones IP públicas a cualquier instancia de su

  cuenta. En lugar de esperar a que un técnico de datos reconfigure o reemplace su host, o bien esperar a que el DNS se propague a todos sus clientes, Amazon EC2 permite solucionar los problemas con su instancia o su software, mediante la reasignación rápida de su dirección Elastic IP a una instancia de sustitución.

- **Amazon Virtual Private Cloud** – Amazon VPC es un puente seguro y sin fisuras entre la infraestructura de TI de una empresa y la nube de Amazon Web Services. Amazon VPC permite a las empresas conectar su infraestructura existente con un conjunto de recursos informáticos aislados de AWS mediante una conexión de VPN (red privada virtual), así como ampliar sus funciones de gestión existentes, como los servicios de seguridad, los cortafuegos y los sistemas de detección de intrusiones para incluir sus recursos de AWS.

- **Amazon CloudWatch** – Amazon CloudWatch es un servicio web que proporciona supervisión para las aplicaciones y los recursos en nube de AWS, empezando por Amazon EC2. Este servicio Web permite visualizar la utilización de recursos, el funcionamiento operativo y los patrones de demanda en general (incluido el uso de CPU, las operaciones de lectura y escritura en disco y el tráfico de red). Puede obtener estadísticas, ver gráficos y definir alarmas para sus datos métricos. Para utilizar Amazon CloudWatch, el usuario simplemente debe seleccionar las instancias de Amazon EC2 que le gustaría supervisar. También puede suministrar sus propios datos métricos empresariales o de

aplicación. Amazon CloudWatch comenzará a agregar y a almacenar los datos de supervisión a los que pueda acceder mediante las herramientas de línea de comandos o las API de servicio web.

- **Auto Scaling** – Auto Scaling permite escalar automáticamente la capacidad de Amazon EC2, para aumentarla o reducirla, de acuerdo con las condiciones que defina. Con Auto Scaling, puede asegurarse de que el número de instancias de Amazon EC2 que esté utilizando aumente sin interrupciones durante los picos de demanda, a fin de mantener el rendimiento, y se reduzca automáticamente durante los períodos de calma en la demanda para minimizar los costes. Auto Scaling resulta especialmente adecuado para aquellas aplicaciones que muestran variaciones de uso según la hora, el día o la semana. Auto Scaling está disponible a través de Amazon CloudWatch y está a su disposición sin ningún pago adicional aparte de las tarifas de Amazon CloudWatch.

- **Elastic Load Balancing** – Elastic Load Balancing distribuye automáticamente el tráfico entrante de las aplicaciones entre varias instancias de Amazon EC2. Permite conseguir aún más tolerancia a fallos en sus aplicaciones, al proporcionar la capacidad de equilibrio de carga necesaria como respuesta al tráfico entrante de aplicaciones. Elastic Load Balancing detecta instancias en mal estado dentro de un conjunto y redirige automáticamente el tráfico hacia las instancias que se encuentran en buen estado, hasta que se restauran las instancias en mal estado. Puede habilitar Elastic Load Balancing en una única Zona de disponibilidad o en varias zonas, para obtener un rendimiento de la aplicación más uniforme. Amazon CloudWatch se puede utilizar para capturar los indicadores operativos de Elastic Load Balancing específico, como el recuento de solicitudes y la latencia de solicitudes, sin ningún coste adicional aparte de la tarifa de Elastic Load Balancing.

- **Clústeres de Computación de alto rendimiento (HPC)** – Los clientes con cargas de trabajo informáticas complejas, como los procesos paralelos estrechamente asociados, o con aplicaciones afectadas por el rendimiento de red, pueden lograr un alto rendimiento informático y de red proporcionado por la infraestructura personalizada y, al mismo tiempo, beneficiarse de la elasticidad,

la flexibilidad y las ventajas de precio de Amazon EC2. Las instancias informáticas en clúster y GPU en clúster se han diseñado específicamente para proporcionar una funcionalidad de red de alto rendimiento y se pueden iniciar, de forma programada en clústeres, lo que permite a las aplicaciones alcanzar el rendimiento de red de baja latencia necesario para la comunicación de nodo a nodo estrechamente asociada. Las instancias informáticas en clúster y GPU en clúster proporcionan también un rendimiento de red mejorado, lo que las hace adecuadas para las aplicaciones personalizadas que necesitan realizar operaciones con un alto consumo de red.

- **VM Import** – VM Import permite importar fácilmente imágenes de equipos virtuales desde el entorno del cliente a instancias de Amazon EC2. VM Import le permite aprovechar sus inversiones en los equipos virtuales que tenga ya

  construidos conforme a sus requisitos de seguridad informática, gestión de configuraciones y cumplimiento normativo. Para ello, traslada impecablemente dichos equipos virtuales a Amazon EC2 para disponer de ellos como instancias listas para su uso. Esta oferta está disponible sin coste adicional, aparte de los cargos por consumo de Amazon EC2 y Amazon S3. Más información sobre VM Import.

2.2.4. Tipos de instancia

Instancias estándar

Las instancias de esta familia suelen resultar adecuadas para la mayoría de las aplicaciones.

- Instancia pequeña (predeterminada) 1,7 GB de memoria, 1 unidad de sistemas EC2 (1 núcleo virtual con 1 unidad de sistemas EC2), 160 GB de almacenamiento de almacenamiento de instancia local, plataforma de 32 o 64 bits

- Instancia mediana 3,75 GB de memoria, 2 unidades de sistemas EC1 (1 núcleo virtual con 2 unidades de sistemas EC2 cada uno), 410 GB de almacenamiento de instancias local, plataforma de 32 o 64 bits
- Instancia grande: 7,5 GB de memoria, 4 unidades de sistemas EC2 (2 núcleos virtuales con 2 unidades de sistemas EC2 cada uno), 850 GB de almacenamiento de instancias local, plataforma de 64 bits
- Instancia extragrande: 15 GB de memoria, 8 unidades de sistemas EC2 (4 núcleos virtuales con 2 unidades de sistemas EC2 cada uno), 1690 GB de almacenamiento de instancias local, plataforma de 64 bits

Microinstancias

Las microinstancias (t1.micro) ofrecen una pequeña cantidad de recursos de CPU consistentes y permiten ampliar la capacidad de CPU en ráfagas cortas cuando haya nuevos ciclos disponibles. Son adecuadas para aplicaciones con una

productividad más baja y sitios web que suelen requerir ciclos de cálculo adicionales con regularidad.

- Micro Instance 613 MB de memoria, hasta ECU (para breves explosiones periódicas), solo almacenamiento EBS, plataforma de 32 bits o 64 bits.

Instancias con gran cantidad de memoria

Las instancias de esta familia ofrecen una memoria de gran tamaño para aplicaciones de alto rendimiento, incluidas las aplicaciones de colocación en caché de memoria y de bases de datos.

- Instancia extragrande con memoria elevada: 17,1 GB de memoria, 6,5 ECU (2 núcleos virtuales con 3,25 unidades de sistemas EC2 cada uno), 420 GB de almacenamiento de instancias local, plataforma de 64 bits

- Instancia extragrande doble con memoria elevada: 34,2 GB de memoria, 13 unidades de sistemas EC2 (4 núcleos virtuales con 3,25 unidades de sistemas EC2 cada uno), 850 GB de almacenamiento de instancias local, plataforma de 64 bits

- Instancia extragrande cuádruple con memoria elevada: 68,4 GB de memoria, 26 unidades de sistemas EC2 (8 núcleos virtuales con 3,25 unidades de sistemas EC2 cada uno), 1690 GB de almacenamiento de instancias local, plataforma de 64 bits

Instancias para CPU de alto rendimiento

Las instancias de esta familia tienen, en proporción, más recursos de CPU que memoria (RAM) y resultan adecuadas para aplicaciones que realizan gran cantidad de sistemas.

- Instancia mediana de CPU elevada: 1,7 GB de memoria, 5 unidades de sistemas EC2 (2 núcleos virtuales con 2,5 unidades de sistemas EC2 cada uno), 350 GB de almacenamiento de instancias, plataforma de 32 o 64 bits

- Instancia extragrande de CPU elevada: 7 GB de memoria, 20 unidades de sistemas EC2 (8 núcleos virtuales con 2,5 unidades de sistemas EC2 cada uno), 1690 GB de almacenamiento de instancias local, plataforma de 64 bits

Instancias informáticas en clúster

Las instancias de esta familia ofrecen, en proporción, recursos de CPU de alto rendimiento y una mejora del rendimiento de red y son adecuadas para aplicaciones de tipo HPC (Informática de alto rendimiento) y otras aplicaciones muy exigentes vinculadas con la red.

- Extra grande cuádruple de sistemas en clúster: 23 GB de memoria, 33,5 de unidades de sistema de EC2, 1690 GB de almacenamiento de instancias local, plataforma de 64 bits, Ethernet de 10 Gigabits

- Extra grande óctuple de sistemas en clúster: 60.5 GB de memoria, 88 de unidades de sistema de EC2, 3370 GB de almacenamiento de instancias local, plataforma de 64 bits, Ethernet de 10 Gigabits

Instancias de GPU para clústeres

Este tipo de instancias ofrece unidades de procesamiento gráfico (GPU) con una CPU proporcionalmente elevada y mejor funcionamiento en red para aplicaciones que se benefician del procesamiento muy paralelizado, incluidas aplicaciones HPC, de representación gráfica o de procesamiento multimedia. Mientras las instancias informáticas en clústeres permiten la creación de clústeres de instancias conectadas mediante una red de baja latencia y altas prestaciones, las instancias GPU en clúster proporcionan una opción adicional para las aplicaciones que pueden beneficiarse de la mayor eficiencia de la potencia de los sistemas informáticos en paralelo que se consigue con las GPU en lugar de con los procesadores tradicionales.

- Extragrande cuádruple con GPU en clúster: 22 GB de memoria, 33,5 unidades de sistemas EC2, 2 GPU NVIDIA Tesla “Fermi” M2050, 1690 GB de almacenamiento de instancias local, plataforma de 64 bits, Ethernet de 10 Gigabits Unidad de sistemas de EC2: Una unidad de sistemas EC2 proporciona la capacidad de CPU equivalente de un procesador Opteron 2007 o Xeon 2007 de 1,0-1,2 GHz.

2.2.5. Sistemas operativos y software.

Sistemas operativos

Las AMI (imágenes de máquina de Amazon) se han preconfigurado con una lista de sistemas operativos cada vez mayor. Amazon trabaja con sus asociados y su comunidad para proporcionar el mayor número posible de opciones. Asimismo, el usuario puede utilizar sus herramientas de empaquetado para cargar sus propios sistemas operativos. Entre los sistemas operativos que puede utilizar actualmente con sus instancias de Amazon EC2 se encuentran los siguientes:

Red Hat Enterprise Linux	Windows Server	Oracle Enterprise Linux
SUSE Linux Enterprise	AMI de Amazon Linux	Ubuntu Linux
Fedora	Gentoo Linux	Debian

Software

Amazon EC2 permite a asociados y clientes crear y personalizar imágenes de máquina de Amazon (AMI) con software según sus necesidades. Amazon tiene cientos de AMI disponibles y pagadas a disposición del cliente. Una pequeña muestra del software disponible para utilizar en Amazon EC2 incluye:

Bases de datos	Gestión de recursos	Alojamiento web
IBM DB2	StackIQ Rocks+	Apache HTTP
IBM Informix Dynamic Server	Hadoop	IIS/Asp.Net
Microsoft SQL Server Standard	Condor	IBM Lotus Web Content Management
MySQL Enterprise		IBM WebSphere Portal Server
Oracle Database 11g		

Entornos de desarrollo de aplicaciones	Servidores de aplicaciones	Codificación y transmisión de vídeos
IBM sMash	IBM WebSphere Application Server	Wowza Media Server Pro
JBoss Enterprise Application Platform	Java Application Server	Windows Media Server
Ruby on Rails	Oracle WebLogic Server	

2.2.6. Precios

La política de AWS es “Pague sólo por lo que utilice”. Los precios listados se basan en la región en la que se ejecuta la instancia.

Las instancias según demanda permiten pagar por la capacidad informática por horas sin compromisos a largo plazo. Esto libera de los costes y las complejidades de la planificación, la compra y el mantenimiento del hardware y transformará lo que normalmente son grandes costes fijos en costes variables mucho más pequeños.

Los precios que aparece más abajo incluye el coste para la ejecución de AMI públicos y privados en el sistema operativo especificado (Los precios de "Uso de Windows" se aplican a Windows Server® 2003 R2, 2008 y 2008 R2). Asimismo, Amazon le ofrece instancias adicionales para Amazon EC2 sobre Microsoft Windows con SQL Server, Amazon EC2 sobre SUSE Linux Enterprise Server, Amazon EC2 sobre Red Hat Enterprise Linux y Amazon EC2 sobre IBM cuyos precios se fijan de forma distinta.

Los precios de la región de Virginia en EE.UU serían, por ejemplo:

	Uso de Linux/UNIX
Instancias según demanda estándar	
Pequeño (Predeterminado)	$0,080 por hora
Mediano	$0,160 por hora
Grande	$0,320 por hora
Extragrande	$0,640 por hora

Microinstancias según demanda	
Micro	$0,020 por hora
Instancias según demanda de alta memoria	
Extragrande	$0,450 por hora
Doble extragrande	$0,900 por hora
Cuádruple extragrande	$1,800 por hora
Instancias según demanda para CPU de alta potencia	
Mediano	$0,165 por hora
Extragrande	$0,660 por hora
Instancias informáticas en clúster	
Cuádruple extragrande	$1,300 por hora
Óctuple extragrande	$2,400 por hora
Instancias de GPU para clústeres	
Cuádruple extragrande	$2,100 por hora

Los precios se calculan por hora de instancia consumida para cada instancia, desde el tiempo en el que se ejecuta hasta que se finaliza. Cada hora de instancia parcial consumida se facturará como hora completa.

3. Startups

3.1. ¿Qué es una startup?

Una startup es una empresa puesta en marcha por algún emprendedor que asocia a la idea empresarial una buena dosis de innovación y la pone en marcha apoyándose en mayor o menor medida en las nuevas tecnologías. Tiene que funcionar bien a pequeña escala y al mismo tiempo demostrar que lo seguirá haciendo a medida que crezca.

3.2. Cómo montar una startup

Uno de los procesos más interesantes en la economía es el nacimiento de una empresa. Los pasos a seguir para poder montar una empresa propia son los siguientes:

1. Encontrar la idea, sin necesidad de que esté perfectamente definida, ya habrá tiempo de afinar cada detalle más adelante.
2. Buscar el equipo, gente que se dedique a tiempo completo a la iniciativa, con habilidades complementarias.
3. Encontrar la financiación. Si se tiene una buena idea y un buen equipo será fácil encontrar financiación. Pero si una de las dos cosas falla es prácticamente imposible.
4. Lanzar la programación y operaciones.

3.2.1. Errores habituales

Empezar escribiendo un plan de negocio. El plan de negocio es importante a la hora de buscar financiación, para explicar a inversores lo que se quiere hacer. Pero más importante, antes que el plan de negocio, es encontrar el equipo con el que montar la empresa. Sin un buen equipo no hay plan de negocio que consiga financiación. Hay muchas empresas que las lanza un emprendedor en solitario. Pero suelen ser empresas que no han necesitado financiación externa. Los inversores lo primero que miran es a quién ha conseguido atraer al proyecto. Tener un equipo de clones. Es sorprendente como muchas start-ups empiezan con un grupo de programadores o un grupo de financieros. Es un gran error. Es necesario que en el equipo fundador haya habilidades complementarias.

3.3. Financiación

Una de las mayores dificultades al montar una startup surge por la falta de financiación, en este apartado hablaremos de los tipos de financiación para las empresas que empiezan y de las opciones de financiación que tiene una startup en España.

Encontrar fondos para la financiación de un proyecto no es tarea fácil por lo que con este proyecto esperamos reducir el problema, haciendo que la cantidad de financiación necesaria disminuya, y así ayudar al nacimiento de nuevas empresas.

3.3.1. Tipos de financiación

A lo largo de la vida de una empresa esta requiere de diferentes inversiones para su crecimiento. Dependiendo de la etapa en la que esté, el tipo de inversión se denomina de forma diferente y tiene unas características particulares. El tipo de inversión suele ir ligado al beneficio de una empresa:

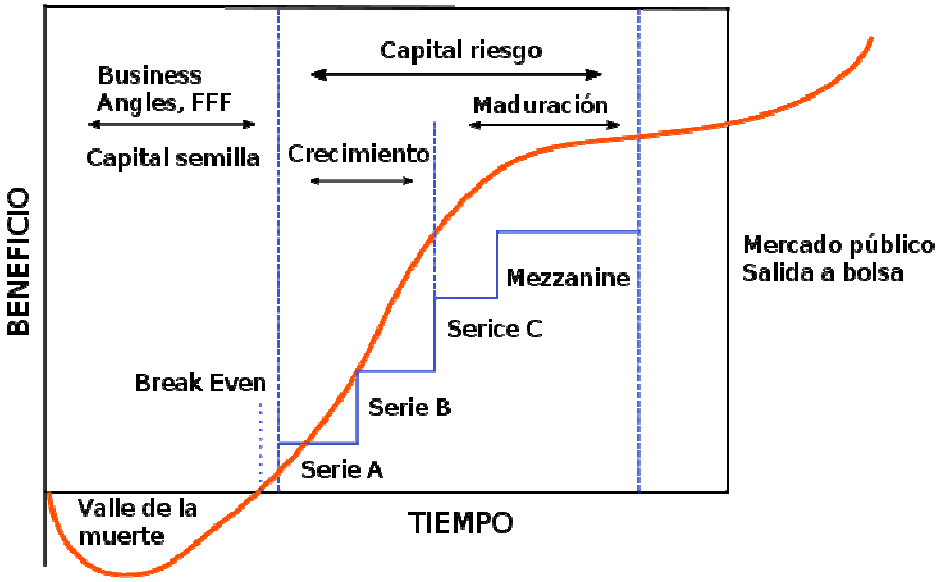


Figura 3.1 Gráfico que muestra los beneficios durante la vida de la empresa.

Para comenzar la actividad, la empresa necesita de un capital que suele salir del bolsillo de los emprendedores y fundadores de dicha empresa. A veces también se pide dinero prestado a la FFF: *"family, friends and fools"*, a la gente cercana a los emprendedores como la familia y los amigos, a este tipo de capital se le puede denominar capital de arranque o capital semilla. Lo normal es que durante el primer año de vida de la empresa esta no genere ningún beneficio, sólo pérdidas. Esta etapa se llama valle de la muerte y es donde suelen morir el 90% de las empresas. Apenas un 5% de las empresas llega a los dos años. Si los emprendedores se saben mover y su idea es buena, pueden tener la suerte de que aparezca un *business angel* que apoye económicamente su idea y se involucre en el proyecto. Normalmente un business angel se involucra cuando la empresa ya ha tenido sus primeras ventas y tiene bastante claro que la empresa va a empezar a ser rentable en poco tiempo. Otras fuentes de financiación en edades tempranas de la empresa suele ser la financiación nacional o regional.

Pasado el *break even*, el punto de inflexión donde los ingresos son superiores a los gastos, suele ser necesaria una inyección de capital y es un buen momento para que el business angel recupere su inversión. A esta nueva inversión se denomina capital privado o *venture capital*. En Estados Unidos el tipo de *venture capital* también se categoriza, y se suele denominar como Serie A, Serie B, Serie C, etc. Cada nueva ronda de financiación para la empresa se pasa a la siguiente letra. La Serie A se caracteriza porque suele ser la primera vez que se ofrece la compañía a inversores externos, como puede ser un business angel con mucho capital o algún grupo de inversión privado. La Serie B suele darse cuando la compañía ya es rentable y se quiere que crezca y se expanda para aumentar el margen de beneficios.

La última gran inversión de la empresa suele ser su salida a bolsa, donde el último inversor recupera su inversión y la compraventa de acciones de la empresa pasa a ser un proceso público donde ya no intervienen los gestores en la decisiones de quien compra y quien vende cada acción de la compañía.

	Inversión	Características
Capital Semilla		• Inicio de la actividad • Fase de desarrollo de productos • Aún no se generan ingresos significativos
Venture Capital Serie A	< 5 m. €	• Compañía genera ingresos aún sin beneficios • Primer vez que la compañía se ofrece a inversores externos • Acciones preferentes
Venture Capital Serie B	5 – 20 m. €	• La compañía ha avanzado en su negocio • Mayor valorización de la compañía
Grow Capital	20 – 250 m. €	• Crecimiento • Expansión
Private Equity	> 250 m.€	• Mezzaine Capital • Empresas en dificultades (suspensión de pagos, quiebra, etc) • Otro tipo de inversiones

3.3.2. Financiación de una startup en España con ayudas públicas

Para buscar financiación, en España hay organismos como el CDTI (con préstamos muy interesantes como el NEOTEC), ENISA o subvenciones públicas como el INNPACTO. También hay firmas privadas de inversión. Y siempre se puede acudir a un banco, todos ofrecen ayudas para emprendedores (por ejemplo la caixa con caixa capital risc).

Las ayudas se pueden clasificar en base a lo avanzado del proyecto:

- *"Tengo una idea en mente y poco más"*: Hasta antes de empezar la crisis había iniciativas que analizaban la viabilidad empresarial de una nueva idea. Por ejemplo en Cataluña el programa Genesi, con el que se podía conseguir hasta 20.000€ a fondo perdido.
- *"Tengo un proyecto en marcha con un pequeño equipo"*: en este caso se puede solicitar un préstamo a ENISA para jóvenes emprendedores o aprovechar también las nuevas líneas de ayudas regionales para fomentar la creación y crecimiento de empresas de base tecnológica, como pueda ser el NEBT en Cataluña.
- *"Tengo una startup con capital semilla"* (fondos propios, family & friends, business angels…): solicita el préstamo NEOTEC al CDTI, siempre que puedas defender que tu proyecto es técnicamente brillante ya que actualmente hay muchísima competencia.

Otra opción es solicitar un préstamo a ENISA, donde existen varias líneas dependiendo del perfil técnico-innovador de tu empresa y que, en función del plan de negocio, se puede financiar como máximo con la misma cantidad aportada por el inversor privado que haya aportado la máxima cantidad individual.

Si se ha formado algún consorcio con PYMEs y/o Universidades para la realización de parte del proyecto, entonces se puede optar también a las líneas de financiación para proyectos colaborativos de CDTI (integrados, cooperación pymes, interempresas). Además si alguno de los partners se encuentra en alguna de las denominadas regiones de "convergencia", el proceso irá mucho más rápido.

- *"Tengo una startup con una primera ronda de financiación privada"* (capital riesgo, capital privado…): solicita el préstamo NOTEC II al CDTI y/o un préstamo a ENISA. No olvides que siempre te favorece más cerrar la ronda privada después de haber negociado las condiciones con los organismos anteriores.

En este caso también aplica lo de las ayudas para la realización de proyectos en colaboración.

- *"Tengo una startup sólida en proceso de crecimiento"*: también existen líneas de financiación atractivas para ti mediante préstamos participativos o capital público, si lo que quieres es acelerar tu crecimiento.

Para todos los casos anteriores existen múltiples ayudas regionales que pueden ser combinadas con las nacionales, aunque algunas pueden llegar a reducir las probabilidades de conseguir futuras ayudas nacionales, por lo que hay que estudiar cada caso antes de aplicar.
Finalmente, siempre y cuando dispongas de paciencia y recursos internos para dedicarse a ello, existen subvenciones para casi todas las facetas de tu negocio: contratación de personal, propiedad intelectual, internacionalización…

3.4. Housing

Hoy en día la mayoría de las empresas disponen de una página web que facilita la captación de clientes y el trato con ellos. Tener una web supone un gasto menor que el de tener un local físico. Es necesario tener un sitio donde alojar nuestra web, para eso existe el Housing.

Los principales clientes de Housing son:

- Empresas de comercio, que utilizan las instalaciones para un ambiente seguro.
- Grandes empresas, que utilizan las instalaciones para evitar desastres usándolas como respaldo de datos fuera del sitio y la continuidad del negocio.
- Compañías de telecomunicaciones, que utilizan las instalaciones para intercambiar tráfico con otras compañías de telecomunicaciones y el acceso a los clientes potenciales.

3.4.1. ¿Qué es el Housing?

Es una modalidad de alojamiento web destinado principalmente a grandes empresas y a empresas de servicios web.

Consiste básicamente en vender o alquilar un espacio físico de un centro de datos para que el cliente coloque ahí su propio ordenador. La empresa le da la corriente y la conexión a Internet, pero el servidor lo elige completamente el cliente, incluso el hardware.

El término "housing" proviene de los países hispanohablantes y otros como Francia. Sin embargo en países de habla inglesa utilizan '*colocation'*, '*colocation centre'* o *también 'Co-Location'.*

Figura 3.2. Ejemplo de instalación de armarios rack.

3.4.2. Instalaciones

Los componentes de una instalación de housing, generalmente, son los rack, ya sea en un 'house blend', donde el proveedor de 'Colocation' es un cliente de los transportistas, y se conecta a sus clientes con su propio router para el acceso a múltiples operadores, o bien el acceso directo 'cross-connect' para los routers de los mismos portadores, o ambos. Además disponen de un sistema de refrigeración, seguridad física (incluida la vigilancia de vídeo, tarjeta de identificación biométrica y el acceso, registro y similares), y el monitoreo en tiempo real de todas estas funciones.

Rack

Un rack es un bastidor destinado a alojar equipamiento electrónico, informático y de comunicaciones. También son llamados bastidores, cabinets o armarios. Los racks son un simple armazón metálico con un ancho normalizado de 19 pulgadas, mientras que el alto y el fondo son variables para adaptarse a las distintas necesidades. El armazón cuenta con guías horizontales donde puede apoyarse el equipamiento, así como puntos de anclaje para los tornillos que fijan dicho equipamiento al armazón. En este sentido, un rack es muy parecido a una simple estantería.

Figura 3.3. Imagen de un Rack.

¿Para qué sirven?

Los racks son muy útiles en un centro de proceso de datos, donde el espacio es escaso y se necesita alojar un gran número de dispositivos. Estos dispositivos suelen ser:

- Servidores cuya carcasa ha sido diseñada para adaptarse al bastidor. Existen servidores de 1U, 2U y 4U, y recientemente, se han popularizado los servidores blade que permiten compactar más de veinte servidores en una altura de 4U, compartiendo fuentes de alimentación y cableado.
- Conmutadores y enrutadores de comunicaciones.
- Cortafuegos.
- Sistemas de audio y video.

El equipamiento simplemente se desliza sobre un raíl horizontal y se fija con tornillos. También existen bandejas que permiten apoyar equipamiento no normalizado. Por ejemplo, un monitor y un teclado.

Edificios

Los edificios que tienen centros de datos dentro de ellos son a menudo fáciles de reconocer debido a la cantidad de equipos de refrigeración ubicados fuera o en el techo.

Las instalaciones de 'Colocation' tienen muchas características especiales:

- Los sistemas de protección contra incendios, incluyendo los elementos de diseño pasivos y activos, así como la implementación de los programas de prevención de incendios en las operaciones. Los detectores de humo se suelen instalar para proporcionar una alerta temprana de un incendio en desarrollo mediante la detección de partículas generadas por combustión lenta componentes anteriores al desarrollo de llama. Esto permite la investigación, la interrupción de la energía, y la extinción de incendios manual de uso de extintores de mano de fuego antes de que el fuego crece hasta un tamaño grande. Un sistema de rociadores contra incendios se proporciona a menudo para controlar un incendio a gran escala si se desarrolla. Limpie los

agentes de los sistemas de extinción de incendios de gases a veces se instala para suprimir un incendio antes que el sistema de rociadores contra incendios. Elementos pasivos de protección contra incendios incluye la instalación de paredes de fuego en todo el espacio, así que un incendio se puede limitar a una parte de la instalación por un tiempo limitado en el caso de que el fracaso de los sistemas de protección activa contra incendios, o si no están instalados.

- Bastidores de 19 pulgadas para equipos de datos y servidores, racks de 23 pulgadas para equipos de telecomunicaciones.

- Los armarios y cajas para control de acceso físico sobre el equipo de los inquilinos.

- Gastos generales o de cables para bastidor de suelo (bandeja) y 'fibreguide', cables de alimentación por lo general en la parrilla por separado de los datos.

- El aire acondicionado se utiliza para controlar la temperatura y la humedad en el espacio. ASHRAE recomienda un rango de temperatura y humedad de las condiciones óptimas de equipos electrónicos frente a los problemas ambientales. La energía eléctrica utilizada por el equipo electrónico se convierte en calor, que es rechazada en el aire ambiente en el espacio de centro de datos. A menos que se elimina el calor, la temperatura ambiente aumentará, lo que resulta en un mal funcionamiento de equipos electrónicos. Mediante el control de la temperatura del aire del espacio, los componentes de servidor en el nivel de la junta se mantienen dentro especificada por el fabricante de temperatura / humedad. Los sistemas de aire acondicionado ayudan a mantener el equipo de humedad espacios dentro de parámetros aceptables por el enfriamiento del espacio aéreo de retorno por debajo del punto de rocío. Demasiada humedad y el agua puede empezar a condensarse sobre los componentes internos. En el caso de una atmósfera seca, los sistemas auxiliares de humidificación puede añadir vapor de agua

en el espacio si la humedad es demasiado bajo, para evitar problemas de descarga de electricidad estática que puede dañar los componentes.

3.4.3. Potencial

Las instalaciones de housing tienen generalmente generadores que se inician automáticamente cuando la energía de la red falla, normalmente se ejecuta con combustible diesel. Estos generadores pueden tener diferentes niveles de redundancia, en función de cómo se construye la instalación.

Los generadores no se inician de forma instantánea, por lo que las instalaciones de 'Colocation' por lo general tienen sistemas de baterías de respaldo. En muchas instalaciones, el operador de la instalación ofrece grandes inversores para proporcionar alimentación de CA de las baterías. En otros casos, los clientes pueden instalar más pequeños UPSes en sus bastidores.

Algunos clientes optan por utilizar el equipo que se alimenta directamente por los bancos de baterías 48VDC (nominal). Esto puede proporcionar una mejor eficiencia energética, y puede reducir el número de piezas que pueden fallar, aunque el voltaje reducido aumenta enormemente la corriente necesaria, y por lo tanto el tamaño (y el coste) de cableado de suministro de energía.

3.4.4. Conexiones

Conexiones Internas

Los propietarios de instalaciones de housing tienen diferentes reglas con respecto a las conexiones cruzadas entre sus clientes, algunos de los cuales

pueden ser portadores. Estas normas pueden permitir a los clientes ejecutar este tipo de conexiones sin cargo, o permitir a los clientes a ordenar este tipo de conexiones de una cuota mensual significativa. Se puede permitir que los clientes ordenen las conexiones cruzadas a los transportistas, pero no a otros clientes.

Algunos centros de ‘Colocation’ tienen un ‘meet-me-room’, donde las diferentes portadoras alojados en el centro pueden intercambiar datos de manera eficiente.

Debido a la alta concentración de servidores dentro de centros de colocación más grandes, la mayoría de las compañías estarán interesadas en la implementación de conexiones directas a estos edificios.

Conexiones externas

Las instalaciones de ‘Colocation’ tienen generalmente varias ubicaciones para cables de fibra óptica para entrar en el edificio, para proporcionar redundancia para que la comunicación puede continuar si un conjunto de cables está dañado. Algunos también tienen conexiones inalámbricas de copia de seguridad, por ejemplo a través de satélite.

3.4.5. Housing en España

El precio medio* de las instalaciones de Housing es España es el siguiente:

- 1/2 armario (22U) → Mensualmente 590€
- Armario completo (42U) → Mensualmente 100 €

Espacio individual

Unidades	Precio
1U	60€
2U	120€
3U	180€
4U	240€
5U	300€

Transferencia

Velocidad	Precio
1Mbps	60€
3Mbps	180€
5Mbps	300€

* Los precios son aproximados. La fuente para tales estimaciones es http://www.ipvedcor.com/housing.html

4.Tecnologías

4.1. Vaadin

Vaadin es un Framework para desarrollar aplicaciones web (RIA – Rich Internet Applications). Cuenta con una arquitectura de servidor, es decir, que la mayor parte de la lógica de la aplicación se ejecuta en el servidor, de forma segura. En la parte del cliente se utiliza Google Web Toolkit (GWT).

Vaadin cuenta con una arquitectura robusta, basada en componentes, junto con el lenguaje Java y las características de enlace de datos ayudarán a construir aplicaciones que sean fáciles de modular y reprogramado.

El núcleo principal del Framework está implementado con Java, y está diseñado para hacer la creación y el mantenimiento de las aplicaciones web fácilmente y con una gran calidad.

La principal idea con el modelo de servidor de Vaadin es que permite olvidarse de la parte del cliente, y programar como cualquier otra aplicación de escritorio Java, lo que te permite además concentrarte en la lógica de la aplicación, no en tecnologías web como HTML o JavaScript.

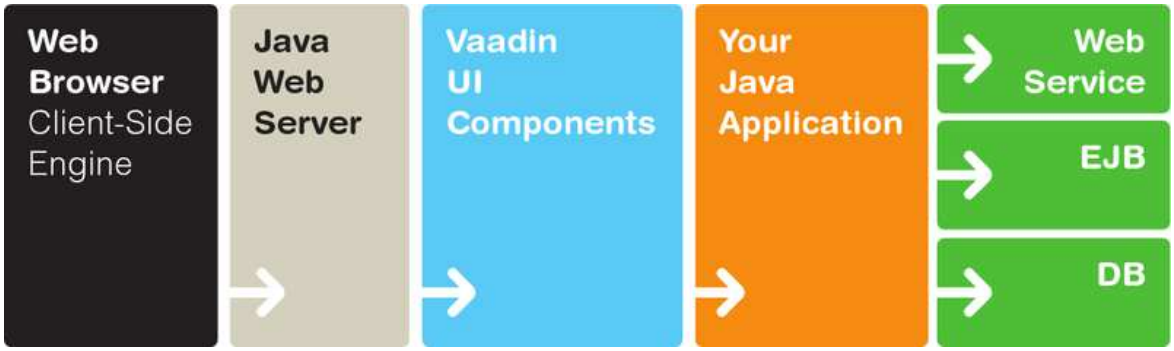


Figura 4.1: Arquitectura básica de las aplicaciones web hechas con Vaadin

Vaadin consiste en el framework del lado servidor, y un motor en el lado cliente que se ejecuta en el navegador como un programa JavaScript, presentando la interfaz de usuario y la entrega de la interacción del usuario con el servidor. La aplicación se ejecuta como una sesión de Java Servlets en un servidor de aplicaciones Java.

Debido a que HTML, JavaScript y otras tecnologías web son invisibles a la lógica de la aplicación, se puede pensar en el navegador simplemente como una plataforma cliente. Éste “ligero” cliente muestra la interfaz de usuario y comunica los eventos del usuario a bajo nivel al servidor. La lógica de control de la interfaz de usuario se ejecuta en un servidor web basado en Java, junto con su lógica de

negocio. Por el contrario, una arquitectura normal cliente-servidor con una aplicación específica en el cliente incluiría una gran cantidad de comunicaciones específicas entre el cliente y el servidor.

Esencialmente, la eliminación de la interfaz de usuario a nivel de la arquitectura de la aplicación hace que éste enfoque sea muy eficaz. Como el motor del lado cliente se ejecuta como JavaScript en el navegador, no son necesarios los plugins del navegador para el uso de aplicaciones hechas con Vaadin. Esto es un filón en los frameworks basados en Flash, applets de Java y otros plugins. Vaadin cuenta con el apoyo de GWT para una amplia gama de navegadores, de modo que el desarrollador no tiene que preocuparse por la compatibilidad con el navegador.

Detrás del modelo de desarrollo basado en el servidor, Vaadin hace el mejor uso de AJAX (*Asynchronous JavaScript and XML*) con técnicas que hacen posible crear aplicaciones RIA (Rich Internet Applications) que son tan sensibles e interactivas como aplicaciones de escritorio.

Bien oculto está el uso de GWT (Google Web Toolkit) en Vaadin, para renderizar la interfaz de usuario en el navegador. Programas GWT están escritos en Java, pero compilados en JavaScript, lo que libera al desarrollador de aprender JavaScript y otras tecnologías web. GWT es ideal para implementar componentes de usuario avanzados (o widgets, en la terminología de GWT) y la lógica de la interacción con el navegador, mientras que Vaadin se encarga de la lógica de la aplicación en el servidor. Vaadin está diseñado para ser extensible, y se pueden utilizar cualquiera de los componentes de GWT fácilmente, además del amplio repertorio ofrecido en Vaadin. El uso de GWT también significa que todo el código que necesita para escribir es Java puro.

4.1.1. Arquitectura de Vaadin

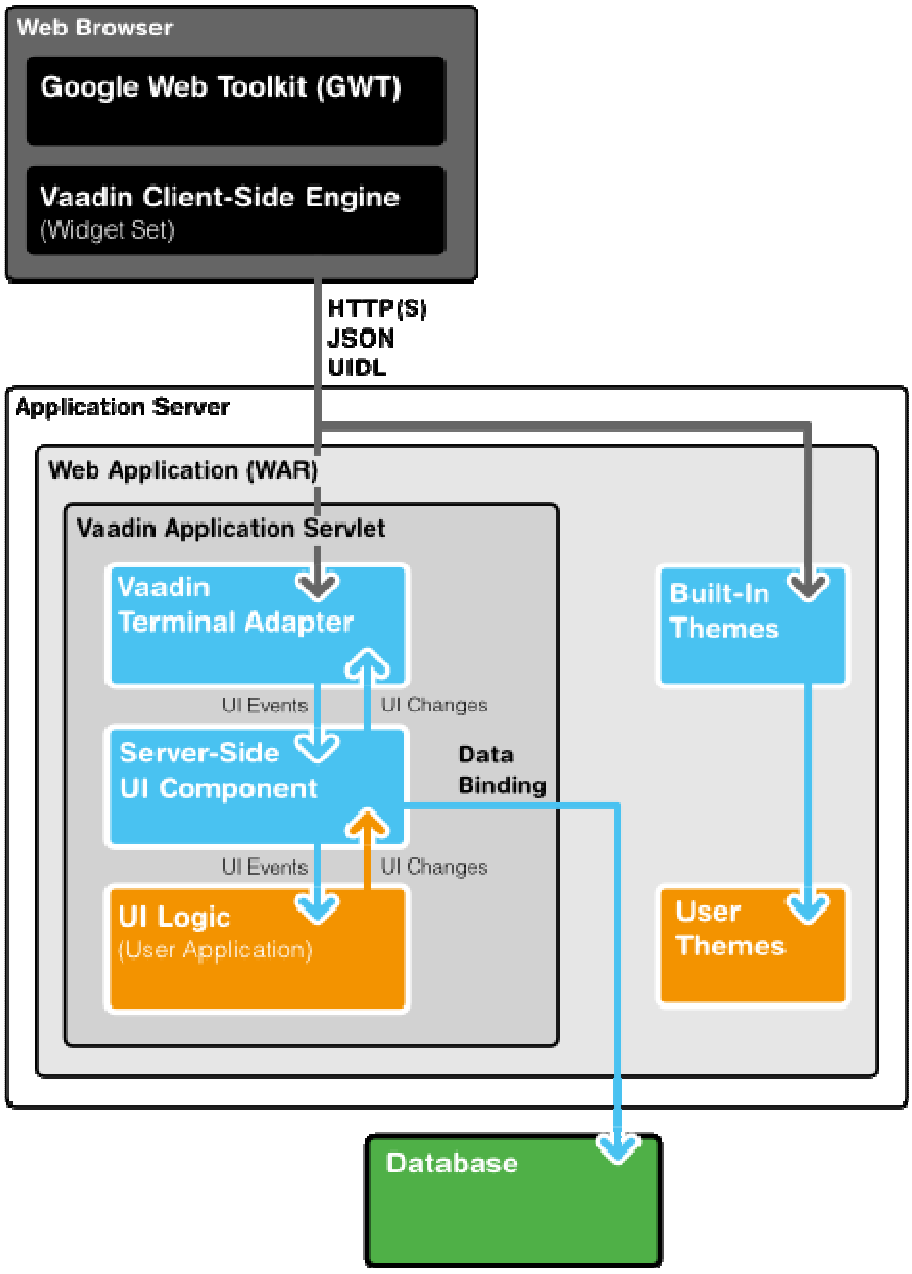


Figura 4.2: Arquitectura de Vaadin

Vaadin consiste en una API para aplicaciones web, una horda de componentes de la interfaz de usuario, temas para el control de la apariencia, y un

modelo de datos que permite el enlace de los componentes de la interfaz de usuario directamente a los datos. Por detrás, también emplea un adaptador de terminal para recibir las solicitudes de los navegadores web y hacer respuestas mostrando las páginas.

Una aplicación que use Vaadin, se ejecuta como un servlet en un servidos Java, proporcionando solicitudes HTTP. El adaptador del terminal recibe solicitudes del cliente a través del servidor Java Servlet, e interpreta los eventos del usuario para una sesión particular. Se realiza un seguimiento de las sesiones con el uso de "cookies". Lo eventos están asociados con los componentes de la interfaz de usuario y son enviados a la aplicación, que se encarga de ellos mediante los "listeners". Si la lógica de la aplicación hace cambios en el lado servidor de los componentes de la interfaz gráfica, el adaptador del terminal genera una respuesta que se representa en el navegador web. El motor del lado cliente que se ejecuta en el navegador recibe las respuestas y las usa para hacer los correspondientes cambios en el navegador web.

El nivel superior de la aplicación se compone de una clase "Aplication" de la que hereda el resto (`com.vaadin.Application`) y que crea los componentes de la interfaz gráfica que se necesitan, recibe sus eventos y hace los correspondientes cambios según esos eventos. Las partes de la arquitectura y sus funciones son las siguientes:

- *Componentes de la interfaz de usuario:*

Consta de componentes que son creados y presentados por la aplicación. Cada componente del lado servidor tiene su "parte contraria" en el lado cliente, con la que el usuario interactúa. Los componentes del lado cliente se pueden serializar sobre la conexión con la parte cliente usando un adaptador de terminal. Los componentes del lado cliente, a su vez, pueden serializar la interacción del usuario con la aplicación, que se reciben en los componentes del lado servidor como eventos. Los componentes retransmiten estos eventos a la lógica de la aplicación. La mayoría de los componente están unidos a una fuente de datos.

- *Motor en el Lado Cliente:*

El motor del cliente de Vaadin controla la visualización en el navegador web usando Google Web Toolkit (GWT). Se comunica la interacción del usuario y los cambios en la interfaz de usuario con el adaptador del terminal del lado servidor, utilizando la "User Interface Definition Language" (UIDL), un lenguaje basado en JSON (*JavaScript object Notation*). Las comunicaciones se realizan mediante el protocolo asíncrono HTTP o HTTPS.

- *Adaptador de Terminal:*

Los componentes de la interfaz de usuario no se visualizan directamente como una página web, por eso es necesario el *"Terminal Adapter".* Esta capa de abstracción permite a los usuarios utilizar aplicaciones Vaadin prácticamente en cualquier navegador web. Versiones 3 y 4 de *"IT Mill Toolkit"* soportan HTML y *simple-AJAX* como base para la visualización, mientras que la versión 5 de Vaadin soporta *advanced-AJAX* como base, mediante el uso de *Google Web Toolkit* (GWT). Se podría, con cualquier tecnología de cualquier navegador, ni siquiera basada en HTML, y se podría hacer funcionar con sólo desarrollar un nuevo adaptador. La aplicación sólo ve la API de Vaadin. Para permitir este tipo de abstracción, los componentes de la interfaz de usuario comunican sus cambios al adaptador, quién los visualiza en el navegador del usuario. Cuando los usuarios hacen algo en la página web, los eventos son comunicados al adaptador terminal, a través del servicio web, como una petición asíncrona AJAX. El adaptador terminal proporciona éstos eventos del usuario a los componentes de la interfaz gráfica, que a su vez se los entrega a la lógica de la aplicación.

- *Temas:*

La interfaz de usuario se divide en presentación y lógica de la aplicación. Mientras que la parte de la lógica es tratada como código Java, la parte de la presentación se define mediante Temas en *CSS*. Vaadin tiene varios temas por

defecto, pero se pueden modificar en las hojas de estilo, añadir imágenes y otros recursos, e incluso se pueden añadir otros temas nuevos definidos por el usuario.

- *UIDL:*

El adaptador de terminal “dibuja” la interfaz de usuario en la página web y cualquier cambio en él se trata con un lenguaje especial (*User Interface Definition Languaje).* Las comunicaciones con éste lenguaje se realizan mediante JSON (*JavaScript object Notation*), que es una forma de intercambiar datos “ligeros” especialmente eficaz y eficiente para interactuar con código JavaScript basado en AJAX en el navegador.

- *Eventos:*

La interacción del usuario con los componente de la interfaz gráfica se realiza con la creación de eventos, que son procesados primero en el lado cliente con JavaScript y después se envían a través del servidor HTTP, adaptador de terminal y las capas de componentes de usuario de la aplicación.

- *Modelo de Datos:*

Además del modelo de la interfaz de usuario, Vaadin proporciona un modelo de datos para interconectar los datos presentados en los componentes de la interfaz de usuario. Usando este modelo de datos, los componentes de la interfaz de usuario puede actualizar los datos de la aplicación directamente, sin la necesidad de usar ningún código de control. Todos los componentes de la interfaz de usuario usan este modelo internamente, pero también se pueden enlazar con cualquier otro modelo de datos. Por ejemplo, se puede enlazar un componente de una tabla con el resultado de una consulta a una base de datos SQL.

4.2. Chef

4.2.1. Introducción

Chef es un framework de integración de sistemas de código abierto construido especialmente para la automatización de la nube. No importa lo complejo que sea el proyecto a desarrollar, Chef hace que sea fácil implementar servidores y aplicaciones de gran escala a lo largo de toda su infraestructura. Debido a que combina los elementos fundamentales de la gestión de la configuración y arquitecturas orientadas a servicios con toda la potencia de Ruby, Chef simplifica la forma de crear una infraestructura elegante y totalmente automatizada.

Figura 4.3: Ilustración del funcionamiento de Chef

Con Chef, se pueden escribir definiciones abstractas como código fuente para describir cómo se quiere que sea cada parte de la infraestructura que se construirá, y luego aplicar esas descripciones a servidores individuales.

El resultado es una infraestructura totalmente automatizada: cuando se trata de un nuevo servidor en línea, lo único que hay que hacer es decirle a Chef cuál es el papel que debe jugar en su arquitectura.

Hablando a alto nivel, Chef es:

- Una librería para la gestión de configuraciones
- Un sistema de gestión de configuraciones
- Una plataforma de integración de sistemas
- Una API para una infraestructura completa

Qué ofrece

Aprovisionamiento

Chef invoca al sistema y llamadas a la API para automatizar el aprovisionamiento de servidores a través de llamadas a la API REST o por medio de la línea de comandos de la interfaz de Knife. En el caso de bare-metal, el chef se integra con cosas como Kickstart para Linux, Jumpstart para Solaris o NIM para AIX. Para los entornos virtualizados, Chef se integra con libvirt y una API hipervisora específica (Xen, KVM, VMWare). En los escenarios de cloud, Chef puede invocar directamente desde la API de las nubes llamadas a los servidores de suministro como AWS y Cloud. Chef también se integra con muchas de las abstracciones de código abierto de lenguajes específicos de cloud como fog para Ruby y jclouds para Java.

Gestión de la Configuración

Chef es una herramienta de gestión de la configuración completamente funcional. Roles y recetas se utilizan para describir cómo se deben configurar los servidores. Chef permite escribir recetas que describen las funciones con las que un servidor debe configurarse de un servidor (como Apache, MySQL, o Hadoop). Estas recetas describen una serie de recursos que deberían estar en un estado en particular, por ejemplo, paquetes que deben ser instalados, los servicios que deben estar en ejecución o archivos que se habrían de escribir. Chef se asegura de que cada recurso está configurado correctamente, sólo tomar acciones correctivas cuando sea necesario. El resultado es un mecanismo seguro y flexible para

asegurarse de que los servidores están siempre funcionando exactamente como el usuario quiere que lo hagan.

.

Integración de Sistemas

Una de las características más potentes del chef está en su diseño ya que separa los datos de configuración de la lógica de configuración. Los datos sobre su infraestructura son almacenados dinámicamente e indexados en una base de datos NoSQL y una se proporciona una API de búsqueda de gran alcance para consultar información acerca de nuestra infraestructura y nuestras aplicaciones. En otras palabras, las recetas de Chef pueden estar basadas en datos proporcionando de este modo la integración del sistema dinámico entre servidores. Por ejemplo, al configurar un servidor web la API de búsqueda puede ser llamada para descubrir la base de datos y servidores memcache y luego actualizar automáticamente la configuración del servidor web. Del mismo modo una receta equilibradora de carga puede añadir automáticamente los servidores web en su configuración.

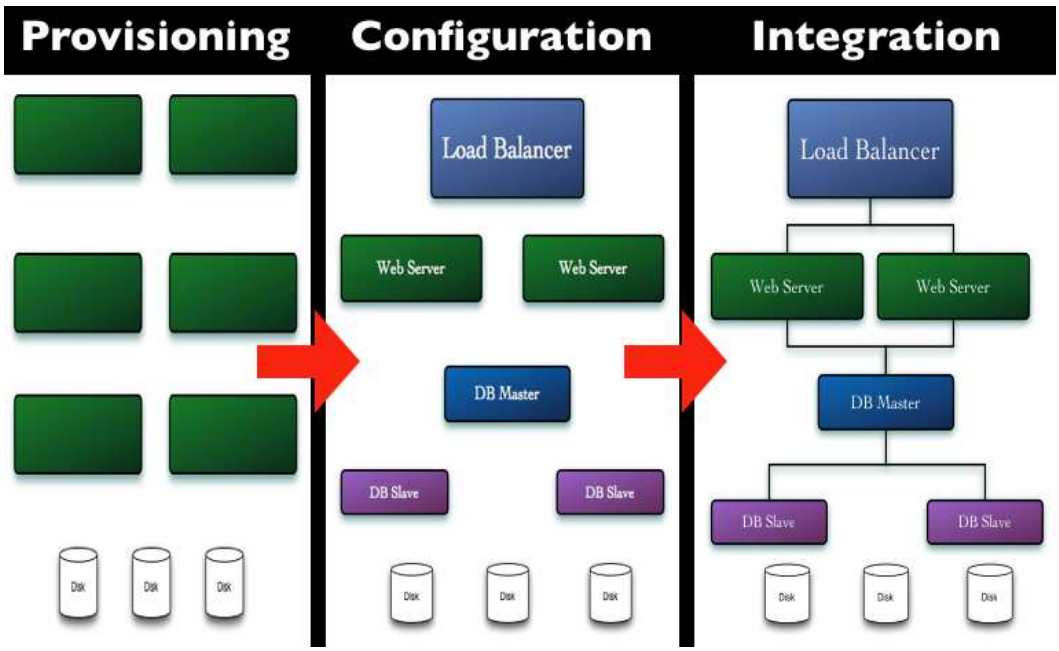


Figura 4.4: Conexión de los recursos de Chef

Aspectos positivos

- *Economía*

Chef le ahorra dinero al usuario al permitir que su negocio pueda soportar infraestructuras más grandes y más complicadas con menos mano de obra, menos repeticiones, y menos errores. Hacer más con menos.

- *Eficiencia*

La automatización de una infraestructura con Chef significa no tener que volver a repetir la configuración de la misma.

- *Escalabilidad*

Chef es escalable horizontalmente como una aplicación web de modo que su crecimiento es rápido.

- *Comodidad*

Para añadir nuevos servidores o modelar nuevos tipos de datos, basta con utilizar la API de Chef y en el momento se puede seguir trabajando sin tener que realizar cambios en el código fuente.

Tipos de Chef

Chef solo

Chef solo es una versión independiente de Chef que se ejecuta localmente en un nodo, separado de un servidor Chef. Esto significa que toda la información y las recetas necesarias para configurar el nodo tienen que estar en el disco local. Estos pueden ser recuperados remotamente a través de una URL, una línea de comandos o el fichero de configuración.

Chef-client and Chef-server

Chef-server es un cliente-servidor que dota a Chef de gran funcionalidad y poder. Chef-client es un agente de Chef que se ejecuta localmente en el sistema (nodo) que estamos gestionando con Chef. Chef-client se conecta con Chef-server para decirle qué hacer en el nodo. Esto permite una gestión de configuraciones más dinámica y flexible. Por ejemplo, los nodos pueden tenener funciones aplicadas a conjuntos de nodos para dar consistencia, pueden consultar información almacenada en el servidor por otros nodos, y configurarse dinámicamente de acuerdo a los cambios en otras partes de la infraestructura.

Hosted Chef

Opscode's Hosted Chef combina la libertad y la flexibilidad de Chef con la fiabilidad y la velocidad de Opscode. Hosted Chef tiene una gran disponibilidad, es escalable con Chef server en la nube y ademas tiene muchas funciones adicionales como el control de acceso de grano fino basado en funciones. También es la primera plataforma de gestión de la configuración como servicio (PaaS), lo que significa que es más fácil empezar usando Chef que gestionar una infraestructura propia. Al igual que con Chef-Server, Chef-cliente se conecta a Hosted Chef para decirle qué hacer en el nodo local: lo que le permite automatizar su infraestructura sin tener que configurar y administrar su propio Chef-Server.

Private Chef

Este Chef es para las empresas que quieren el poder, la flexibilidad, la disponibilidad y el rendimiento de Hosted Chef, pero requieren que la información nunca salga de sus redes privadas.

4.2.2. Arquitectura

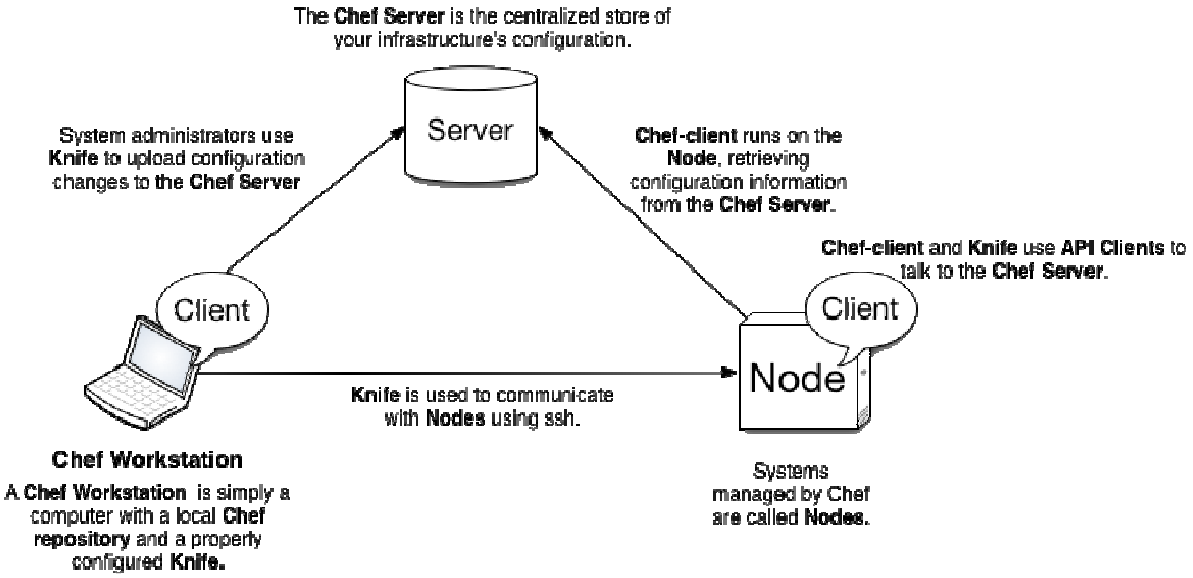


Figura 4.5: Arquitectura de Chef

Al usar Chef para gestionar una infraestructura hay que trabajar con tres tipos de hosts: Chef server, nodos y Chef Workstations.

Chef Server

Este es el sitio donde se almacenan los datos de configuración de una infraestructura.Chef Server almacena los datos necesarios para configurar los nodos y proporciona búsqueda, una herramienta poderosa que le permite manejar de forma dinámica la configuración del nodo basada en datos. Una REST API hace accesibles estos datos accesibles desde los nodos y la Chef Workstation. Opcionalmente se puede usar la Chef Server's WebUI la gestionar la infraestructura mediante una interfaz web. Los usuarios de Hosted Chef tienen un Chef Server gestionado por Opscode como un servicio con una interfaz Web.

Nodos

Un nodo es cualquier host configurado usando Chef-client. Chef-client se ejecuta en los nodos contactando con el Chef Server para obtener la información necesaria para configurar el nodo. Desde el punto de vista de Chef Server, un nodo no es más que una lista de ejecutables, una lista de recetas y funciones que será aplicada sobre el nodo y definirá su configuración y atributos, un conjunto de datos del propio nodo. Como un nodo es una máquina que ejecuta el software de Chef-Cliente, los nodos pueden ser referidos como “clientes”. Esta forma de referenciar los nodos implica la unión entre el ejecutable de Chef-client y la API cuya llamada necesita una autenticación y autorización proporcionadas por la información de identidad del cliente objeto.

Chef Workstations

Una Chef Worksation es el host que cada usuario utiliza para modificar sus cookbooks y otras configuraciones típicas del administrador local. Tiene dos componentes clave:

- Knife, que es un ejecutable incluido en Chef.
- Un repositorio que contiene los documentos de configuración de la infraestructura.

Como ya se ha visto, estos documentos de configuración incluyen cookbooks, datos, funciones y muchas otras cosas. Estos documentos de configuración estan gestionados por un sistema de control de versiones. Usando Knife, la estación de trabajo se usa para actualizar los datos de configuración de Chef Server y para comunicar nodos individuales mediante SSH cuando es necesario.

4.2.3. Conceptos básicos

- *Atributos*:

Los atributos son datos de los nodos como la dirección IP, el nombre del host, los módulos del Kernel cargados, la versión de los lenguajes de programación disponibles en el sistema y mucho más. Durante la ejecución de Chef, Chef Cliente guarda estos atributos en el Chef Server donde son indexados para las búsquedas. Cuando Chef Client se ejecuta otra vez, se recuperan los atributos que fueron guardados anteriormente y los junta basándose en unas reglas de prioridad.

- *Cookbooks:*

Son unidades fundamentales de Chef. Encapsulan todos los recursos que se necesitan para crear una infraestructura y para facilitar que puedan ser compartidas con otros usuarios de Chef. Los cookbooks contienen:

  - Atributos que configuran los valores por defecto utilizados en cualquier parte del cookbook.
  - Definiciones que permiten crear colecciones de recursos reutilizables.
  - Estructuras de archivos que son transferidas a las maquinas administradas por Chef.
  - Librerías que extienden Chef.
  - Recetas que especifican los recursos para la gestión.
  - Recursos ligeros y los proveedores (LWRP) que permiten crear proveedores y recursos personalizados.
  - Plantillas de configuración que están integradas en Chef con valores sustituibles de forma dinámica.
  - Metadatos que le dicen a Chef cuáles son las dependencias entre las recetas, la versión que contienen, las plataformas compatibles y otros datos de interés.

- *Data Bags o paquetes de datos*:

Proporcionan almacenamiento de datos JSON disponibles globalmente. Estos paquetes de datos están almacenados en un directorio en la máquina que está ejecutando Chef-solo. Las recetas pueden cargar Data bags directamente o buscar un paquete de datos para valores específicos de los atributos de los nodos.

- *JSON (JavaScript Object Notation)*:

Es un formato ligero de datos fácil de escribir y leer. Todas las APIS usadas en Chef están hechas con datos JSON. Está construido en dos estructuras:

  - Una colección de pares nombre/valor.
  - Una lista ordenada de valores.

- *Búsqueda:*

Es una característica de Chef Server que permite consultar información sobre la infraestructura y las aplicaciones. Este servicio se puede usar en una receta o a través del comando de búsqueda de Knife.

- *Knife:*

Es una potente interfaz por línea de comandos integrada en Chef. Interactúa con Chef Server a través de la misma REST API que usa el software de Chef Client. Está implementado con una serie de subcomandos que proporcionan funcionalidad para específicas acciones sobre cookbooks, nodos, etc en la propia infraestructura.

- *Role, función u oficio:*

Proporciona una forma de agrupar características de nodos similares facilitando la composición de paquetes de funcionalidad. Los roles están formados por atributos y por una lista de ejecución. Los nodos pueden tener varios roles aplicados proporcionando una lista de recetas para ese nodo. Cuando Chef-cliente se ejecuta fusiona sus propios atributos y ejecuta la lista de los roles que tiene asignados.

4.3. Otros

4.3.1. MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Se ha convertido en el gestor de bases de datos de código abierto más popular debido a su alto rendimiento, alta fiabilidad y facilidad de uso. También es la elección para una nueva generación de aplicaciones basadas en LAMP (Linux, Apache, MySQL, PHP / Perl / Python). MySQL se ejecuta en más de 20 plataformas, incluyendo Linux, Windows, Mac OS, Solaris, AIX de IBM.

Características:

- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación
- Búsqueda e indexación de campos de texto.
- Permite escoger entre múltiples motores de almacenamiento para cada tabla. En MySQL 5.0 éstos debían añadirse en tiempo de compilación, a partir de MySQL 5.1 se pueden añadir dinámicamente en tiempo de ejecución:
    - Los hay nativos como MyISAM, Falcon, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example
    - Desarrollados por *partners* como solidDB, NitroEDB, ScaleDB, TokuDB, Infobright (antes Brighthouse), Kickfire, XtraDB, IBM DB2). InnoDB Estuvo desarrollado así pero ahora pertenece también a Oracle
    - Desarrollados por la comunidad como memcache, httpd, PBXT y Revision

- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

4.3.2. Jboss

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc., adquirida por Red Hat en abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El proyecto está apoyado por una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios. JBoss implementa todo el paquete de servicios de J2EE.

JBoss es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss puede ser descargado, utilizado, incrustado y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

Las características destacadas de JBoss incluyen :

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX

¿Qué ofrece JBoss?

EJB 3.0:

Implementa la especificación inicial de EJB 3.0.

JBoss AOP:

JBoss AOP está orientado a trabajar con Programación Orientada a Aspectos. Esto permitirá añadir fácilmente servicios empresariales (transacciones, seguridad, persistencia) a clases Java simples.

Hibernate:

Es un servicio de persistencia objeto/relaciones y consultas para Java. Hibernate facilita a los desarrolladores crear las clases de persistencia utilizando el lenguaje Java - incluyendo la asociación, herencia, polimorfismo y composición y el entorno de colecciones Java.

JBoss Cache:

Es un producto diseñado para almacenar en caché los objetos Java más frecuentemente accedidos de manera que aumente de forma notable el rendimiento de aplicaciones e-bussines. Eliminando accesos innecesarios a la base de datos, JBoss Cache reduce el tráfico de red e incrementa la escalabilidad de las aplicaciones.

Proporciona dos APIs de caché que se ajustan a nuestras necesidades. La API de JBossCache ofrece una caché tradicional basada en nodos y estructurada en árbol, y la API JBossCacheAOP, edificada sobre la API de JBossCache, proporciona capacidad para la replicación de objetos Java de grano fino, con el máximo beneficio del rendimiento.

JBoss IDE:

Brinda una IDE Eclipse para el JBoss AS. De esta forma la depuración y otras tareas asociadas al desarrollo de aplicaciones puede ser realizadas desde el entorno de Eclipse.

JBoss jBPM:

Gestor de procesos de negocio, también denominado "WorkFlow". jBPM es una plataforma para lenguajes de procesos ejecutables, cubriendo desde gestión de procesos de negocio (BPM) bajo workflow hasta orquestación de servicios. Actualmente jBPM soporta tres lenguajes de procesos, cada uno enfocado a un ambiente y funcionalidad especifica:

- jPDL
- BPEL
- Pageflow

jBPM soporta a estos lenguajes de procesos sobre una sola tecnología: Máquina Virtual de Procesos(PVM)

JBoss Portal:

Es una plataforma de código abierto para albergar y servir una interfaz de portales Web, publicando y gestionando el contenido así como adaptando el aspecto de la presentación.

4.3.3. iBATIS

iBATIS es un “framework de persistencia” que automatiza la relación entre las bases de datos SQL y objetos en Java, .Net o Ruby On Rails. Desarrollado por Apache Software Foundation, es de código abierto, y está basado en capas. Como hemos dicho, se encarga de la persistencia (pues se sitúa entre la lógica de Negocio y la capa de la Base de Datos).

iBATIS asocia objetos de modelo (JavaBeans) con sentencias SQL o procedimientos almacenados mediante ficheros descriptores XML, simplificando la utilización de bases de datos, y tiene como resultado una reducción significativa en la cantidad de código que un desarrollador necesita para acceder a una base de datos relacional usando las APIs de bajo nivel como pueden ser JDBC y ODBC.

La capa de persistencia se configura mediante un fichero XML de configuración, sql-map-config.xml y cada objeto de modelo, que representa al objeto

en la aplicación, se relaciona con un fichero del tipo sqlMap.xml, que contiene sus sentencias SQL. Por ejemplo, un objeto Java *Usuario* se representa con un objeto XML *usuario.xml*.

Es posible subdividir la capa de Persistencia en tres subcapas:

- La capa de Abstracción será la interfaz con la capa de la lógica de negocio, haciendo las veces de "facade" entre la aplicación y la persistencia. Se implementa de forma general mediante el patrón Data Access Object (DAO), y particularmente en iBATIS se implementa utilizando su framework DAO (ibatis-dao.jar).
- La capa de Framework de Persistencia será la interfaz con el gestor de Base de Datos ocupándose de la gestión de los datos mediante un API. Normalmente en Java se utiliza JDBC; iBATIS utiliza su framework SQL-MAP (ibatis-sqlmap.jar).
- La capa de Driver se ocupa de la comunicación con la propia Base de Datos utilizando un Driver específico para la misma.

Toda implementación de iBATIS incluye los siguientes componentes:

- Data Mapper: proporciona una forma sencilla de interacción de datos entre los objetos Java y .NET y bases de datos relacionales.
- Data Access Object: abstracción que oculta la persistencia de objetos en la aplicación y proporciona un API de acceso a datos al resto de la aplicación

5. Constructor

5.1. Introducción

Siempre ha habido gente emprendedora con buenas ideas sin miedo a arriesgarse para conseguir el éxito final, gente que monta su propia empresa y se lanza al mercado llenándolo de innovadoras ideas. Debido a la situación económica actual el número de statups que aparecen cada año ha ido decreciendo lo cual no ayuda a que mejore la economía entrando así en un bucle sin fin.

Este es uno de los principales motivos por los que se decidió llevar a cabo el desarrollo de este proyecto. Constructor pretende ser una herramienta de ayuda para las startups, haciendo que los empresarios emprendedores puedan ahorrar dinero en sus instalaciones hardware y software ya que pueden tener muchos ordenadores diferentes con características diferentes accesibles desde cualquier parte, sin ningún tipo de gasto de mantenimiento y sin el riesgo de obsolescencia propio de los elementos hardware.

Apoyándonos en el auge de la computación cloud, en los grandes avances en este campo y en las facilidades que proporciona hemos desarrollado una aplicación que permite, de forma sencilla, gestionar máquinas virtuales con configuraciones personalizadas y con un gasto mínimo proporcional al número de horas de uso de estas máquinas.

5.2. Planificación

5.3. Requisitos y casos de uso

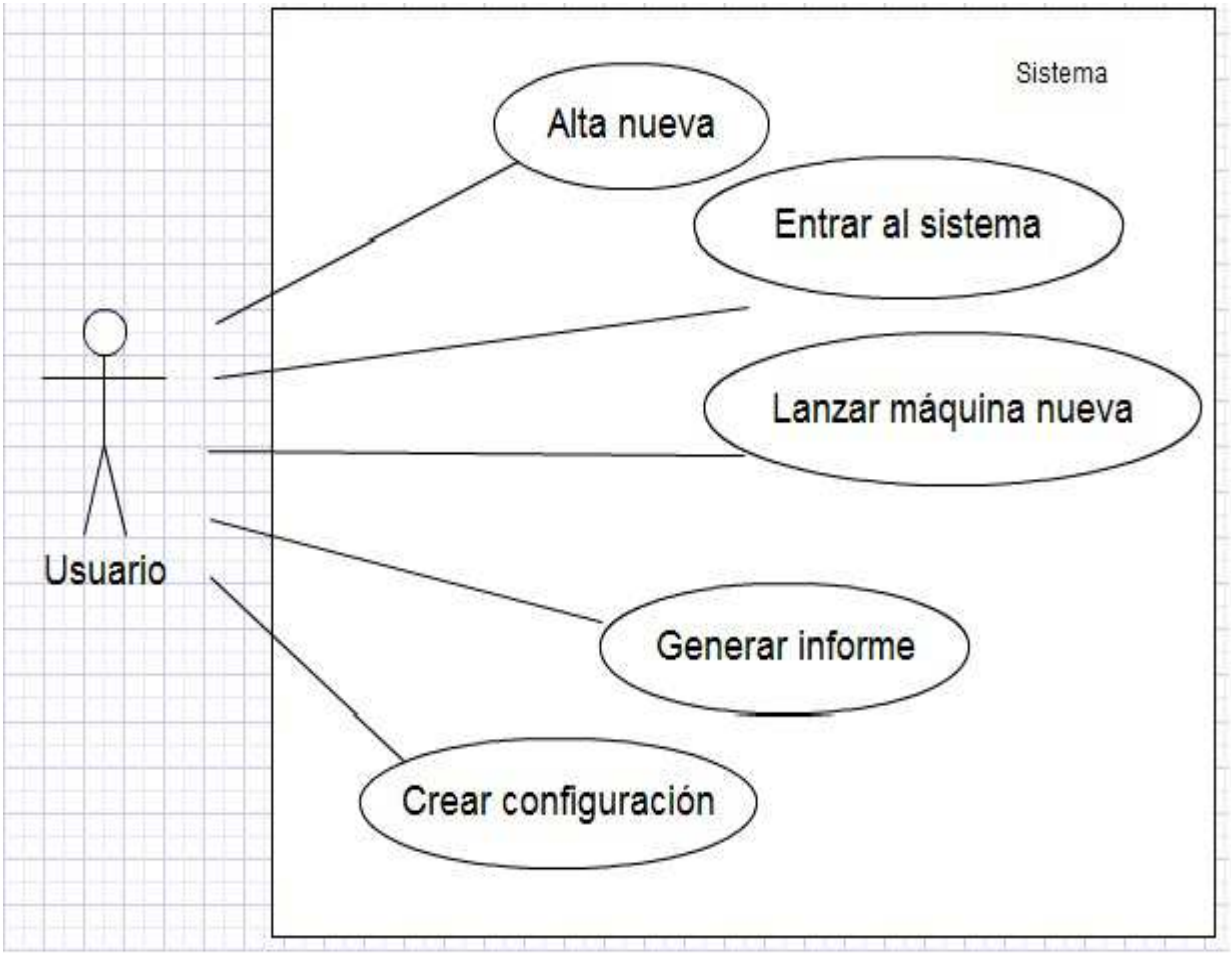
5.3.1. Requisitos del sistema

Un requisito es una condición que debe cumplir un sistema para satisfacer una norma o contrato, en nuestro caso, nos impusimos una serie de requisitos como objetivos para conseguir que Constructor hiciera lo que queríamos que hiciera.

- **Requistitos funcionales**
  - *Lanzar máquinas:* El sistema tiene que ser capaz de lanzar máquinas virtuales con la configuración elegida por el usuario.
  - *Crear ficheros de configuración:* El sistema tiene que ser capaz de crear ficheros de configuración con las características especificadas.
  - *Generar informe:* El sistema tiene que ser capaz de generar un informe con los datos de las máquinas de cada usuario, en el que se refleje el consumo.
- **Requistos no funcionales**
  - *Facilidad de uso:* El usuario tiene que ser capaz de utilizar la aplicación sin recibir entrenamiento.
  - *Seguridad:* Control de accesos, cada usuario tiene que tener sus propios identificador y palabra clave para acceder al sistema
  - *Soporte:* El sistema debe ser instalable por los usuarios.
    El servidor tiene que ser Linux.
    El cliente tiene que tener un navegador IE6 o superior.

5.3.2. Casos de uso

Los casos de uso capturan el comportamiento deseado del sistema definido en los requisitos funcionales y reflejan las interacciones entre el usuario y el sistema.



Caso de uso 1: Alta nueva

- *Precondición*: El usuario desea registrarse en la base de datos.
- *Secuencia normal*
  1. El usuario introduce su nombre de usuario y contraseña
  2. El sistema guarda los datos en la base de datos.
- *Excepciones:*
  2. El nombre de usuario ya está en la base de datos, el sistema muestra un mensaje de error. El caso de uso termina.
- *Postcondición:* El usuario queda registrado en la base de datos.

Caso de uso 2: Entrar al sistema

- Precondición: El usuario desea entrar a la aplicación.
- Secuencia normal:
  1. El usuario introduce su nombre de usuario y contraseña
  2. El sistema da acceso a la aplicación mostrando la página principal.
- Excepciones:
  2. El nombre de usuario o la contraseña son incorrectos. El sistema muestra un mensaje de error. El caso de uso termina.
  2. El nombre de usuario no está en la base de datos. El sistema muestra un mensaje de error. El caso de uso termina.

Caso de uso 3: Lanzar máquina nueva

- Precondición: El usuario quiere crear una máquina nueva.
- Secuencia normal:
  1. El usuario introduce el nombre de la máquina, selecciona la AMI y elige el tamaño.
  2. El sistema conecta con la API de Amazon y lanza una máquina nueva con ese tamaño.
  3. El sistema registra la nueva máquina en la base de datos.
- Excepciones:
  2. Alguno de los campos no ha sido rellenado correctamente. El sistema muestra un mensaje de error. El caso de uso termina.
  3. El usuario ya tiene una máquina con ese nombre introducido. El sistema muestra un mensaje de error. El caso de uso termina.
- Postcondición: La máquina virtual está lanzada y registrada en la base de datos.

Caso de uso 4: Generar informe

- Precondición: El usuario tiene máquinas virtuales lanzadas
- Secuencia normal
  1. El usuario pulsa el botón “generate & report”.
  2. El sistema consulta la base de datos y genera el informe.
  3. El sistema solicita al usuario una carpeta destino
  4. El usuario indica la carpeta destino.
  5. El sistema guarda el fichero en la carpeta indicada.
- Excepciones:
  5. El usuario no ha indicado ninguna carpeta destino. El sistema muetra un mensaje de error, y vuelve al paso 3.

Caso de uso 5: Crear configuración

- Precondición: La máquina virtual que se quiere configurar existe
- Secuencia normal:
  1. El usuario selecciona las características que quiere que tenga su máquina.
  2. El sistema genera los scripts de configuración y los lanza remotamente
- Postcondición: La máquina virtual queda configurada con los paquetes elegidos.

5.4. Arquitectura

En este apartado se detallan las características arquitectónicas y funcionales del conjunto deherramientas que forman Constructor.

5.4.1. Estructura

Debido a la naturaleza de Vaadin, nuestra aplicación web está construida mediante un modelo cliente-servidor, donde toda la lógica de la aplicación reside en el servidor, y en el cliente únicamente se visualizan las ventanas y demás información.

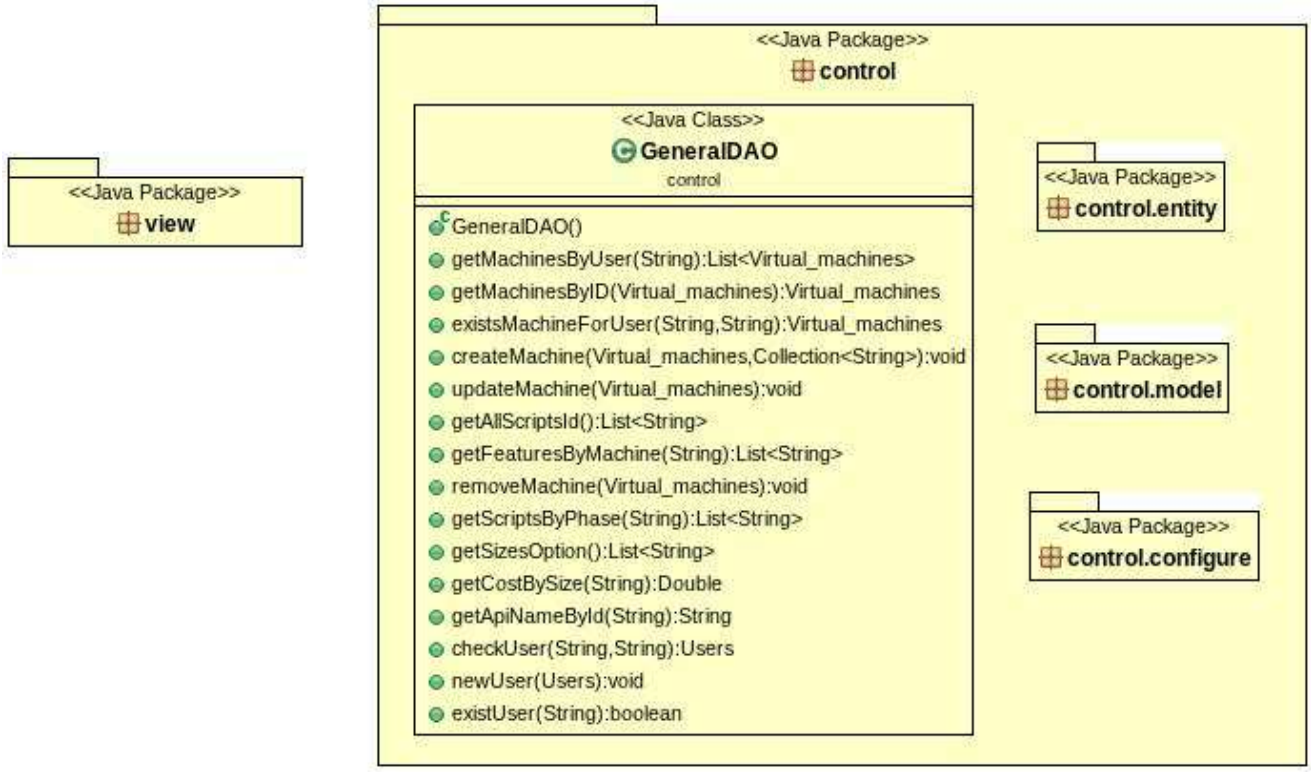
Esto se refleja en la distribución de las clases mediante paquetes, la aplicación está compuesta por dos paquetes principales, “view” que se corresponde con el cliente y “control” que corresponde al servidor.

Las clases que se encargan de la lógica de la aplicación están el paquete "control", dividido en "configure" para las clases de configuración y gestión de las máquinas virtuales, y "entity" para los DAOs y demás clases necesarias para la conexión con la base de datos. Por otro lado se encuentra el paquete "view" que contiene las clases para de visualización. Una de las funciones del paquete "control" es conectar la aplicación con la base de datos, esto lo hacemos mediante la clase "GeneralDAO" la cual contiene las consultas que se hacen a la base de datos utilizando los "DAOS" que genera iBATIS, hay uno para cada tabla de la base de datos y están guardados dentro del paquete "model".

El paquete "map" contiene los ficheros .xml necesarios para la conexión con la base de datos. "Control" tambien se encarga de lanzar y configurar las maquinas, lo primero lo hace la clase "ManageAws" y lo segundo el paquete chef cuya clase

principal es “ConfVM”, el trabajo de estas clases se explicará mas adelante en este mismo capítulo.

Como ya se ha mencionado el paquete “view” es el que se encarga de la parte gráfica, “MainWindow” es la clase principal del paquete, la que controla las demás. “LoginView”, “NewView” y “ListView” implementan una ventana diferente cada una, “GenerateReport” es la encargada de generar los resúmenes de gastos de las máquinas de cada usuario.



Para conseguir el funcionamiento de la aplicación ha sido necesario incorporar diversas librerías específicas.

Entre las más importantes cabe destacar la de AWS usada para lanzar máquinas virtuales y vaadin para construir toda la interfaz gráfica. Gson es la que nos permite personalizar los scripts para la configuración. Ibatis y mysql son las que generan los DAOS y se conectan con la base de datos y poi es la librería que

permite generar los informes en formato .xls . A parte tenemos la librería fileSystem que nos permite hacer operaciones con archivos como copiarlos muy fácilmente.

5.4.2. Base de Datos

La aplicación tiene una base de datos creada en XAMPP usando MySQL. La base de datos es muy sencilla, está formada por 5 tablas:

- Users: almacena la información de los usuarios registrados en dos columnas: username que contiene el nombre del usuario y password que guarda la contraseña de dicho usuario, debido a las restricciones impuestas en nuestra base de datos no puede haber dos usuarios con el mismo nombre.
- Virtual_machines: es la tabla más importante de la base de datos, guarda la información de las máquinas virtuales, su identificador, la AMI utilizada para crearla, el usuario que la ha creado, la fecha, el DNS y el tamaño entre otras cosas. Un usuario no puede tener dos máquinas con el mismo nombre.
- Scripts: esta tabla almacena las configuraciones disponibles en el sistema, y si es necesario, el “paquete” al que pertenece la configuración.
- Features: Relaciona las máquinas con las configuraciones que tiene.
- Size: almacena los distintos tamaños que puede tener una máquina virtual en AWS, así como el coste de cada una y el nombre de la API de AWS.

La aplicación se conecta con la base de datos mediante la librería mysql-connector-java-5.1.6-bin.jar, y generamos los DAOs (Data Access Objects) y todos los Beans mediante el Framework Ibatis de Apache, que además genera todas las consultas necesarias a la base de datos de manera automática.

La clase en la que se describe el enlace entre la base de datos y la aplicación es la siguiente:

SqlMap.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMapConfig

 PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"

 "/WEB-INF/sql-map-config-2.dtd">

<sqlMapConfig>

    <settings useStatementNamespaces="true" />

    <transactionManager type="JDBC" commitRequired="true">

        <dataSource type="JNDI">

            <property name="DataSource" value="java:constructor-ds" />

            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver" />

            <property name="JDBC.ConnectionURL" value="jdbc:mysql://localhost/constructor-ds" />

            <property name="JDBC.Username" value="" />

            <property name="JDBC.Password" value="" />

        </dataSource>

    </transactionManager>

    <sqlMap resource="control/model/map/features_SqlMap.xml" />

    <sqlMap resource="control/model/map/scripts_SqlMap.xml" />

    <sqlMap resource="control/model/map/users_SqlMap.xml" />

    <sqlMap resource="control/model/map/virtual_machines_SqlMap.xml" />

    <sqlMap resource="control/model/map/size_SqlMap.xml" />

</sqlMapConfig>
```

Que como vemos indica el Driver con que conectarse a la base de datos y las tablas que contiene la misma.

5.5. Funcionamiento de la Aplicación

5.5.1. Configuración de las máquinas de AWS

Chef permite escribir recetas que describen las funciones con las que un servidor debe configurarse (como Apache, MySQL, o Hadoop).

Gracias a Chef-solo, a través de un fichero, podemos configurar a nuestro gusto las máquinas de AWS, sin necesidad de instalar nada en ninguna de ellas, únicamente llamando a un script que se encarga de enviar los paquetes necesarios a la máquina virtual e instalarlos en la misma.

deploy.sh: Comienza el proceso, y envía los paquetes a la máquina virtual:

```
#!/bin/bash –x
# deploy.sh [solo] [host]
solo="$1"
host="$2"
tar cj . | ssh -o 'StrictHostKeyChecking no' -i cdhit-key.pem "$host" '
sudo rm -rf ~/chef &&
mkdir ~/chef &&
cd ~/chef &&
tar xj &&
sudo bash install.sh '$solo' '
```

intall.sh :Script que se encarga de la instalación en la máquina remota:

```
#!/bin/bash
# This runs as root on the server
chef_binary=/var/lib/gems/1.9.1/gems/chef-0.10.0/bin/chef-solo
solo=solo1.json
for param in "$@" do  solo="$param"
done
if ! test -f "$chef_binary"; then
        export DEBIAN_FRONTEND=noninteractive
        sudo aptitude update && apt-get -o Dpkg::Options::="--force-confnew"
        --force-yes -fuy dist-upgrade && # Install Ruby and Chef
        sudo aptitude install -y ruby1.9.1 ruby1.9.1-dev make &&
         sudo gem1.9.1 install --no-rdoc --no-ri chef --version 0.10.0
fi &&
"$chef_binary" -c solo.rb -j "$solo"
```

solo.rb : Archivo que indica dónde se encuentran las recetas

```
root = File.absolute_path(File.dirname(__FILE__))
cookbook_path root + '/cookbooks'
```

Y un archivo “solo.json” que indica las recetas a instalar en la máquina remota, y que se genera automáticamente según la configuración seleccionada por el usuario, este es de la forma:

```
{
"run_list": [ "recipe[aws::default]" ,
"recipe[apache2::default]"]
}
```

La configuración de las máquinas virtuales mediante “cookbooks” de Chef se realiza desde la clase *ConfigMV* que se encarga de generar el fichero solo.json en función de las peticiones del usuario y una vez terminada la creacin de este, procede a lanzar los scripts de configuración a la máquina remota, definidos previamente.

El proceso empieza creando el fichero “solo.json” con la configuración seleccionada por el usuario, que lo recibe como parámetro de entrada, y usando la librería Gson genera el archivo “.json”

```
Solo solo1 = new Solo(selectedConf);
 Gson gson = new Gson();
 String json = gson.toJson(solo1);
```

Después se copian todos los scripts necesarios a una carpeta temporal única en el servidor en el que está desplegada la aplicación. La carpeta donde se copian estos archivos suele ser “/temp”, aunque depende del servidor de aplicaciones utilizado. Gracias a “createTempFile” podemos crear una carpeta temporal para cada cliente, con un número de identificación único para permitir la concurrencia en el sistema.

```
File dirTemp = File.createTempFile("chefDir", "");
String chefDir = dirTemp.getAbsolutePath();
 dirTemp.delete();
Process pr1 = Runtime.getRuntime().exec("mkdir " + chefDir);
  pr1.waitFor();
```

El archivo “package.zip” contiene las recetas y otros archivos necesarios para la configuración. Lo copiamos en la carpeta temporal que acabamos de crear, lo descomprimimos y posteriormente borramos el archivo comprimido.

```
 InputStream inP = ConfVM.class.getResourceAsStream("package.zip");
File zipTemp = File.createTempFile("package", ".zip", new File(chefDir));
OutputStream outP = new FileOutputStream(zipTemp);
 inP.close();
outP.flush();
outP.close();
FileSystem.copy(inP, outP);
Process pr2 = Runtime.getRuntime().exec("unzip " + zipTemp.getAbsolutePath() + " -d "
                + zipTemp.getParent());
pr2.waitFor();
zipTemp.delete();
```

A continuación se crean en la carpeta temporal el script deploy.sh y soloConf.json

```
File tempSolo = File.createTempFile("soloConf", ".json", new File(chefDir));
File tempDeploy = File.createTempFile("deploy", ".sh", new File(chefDir));
 PrintWriter writer = new PrintWriter(tempSolo)
 writer.print(json);
 writer.close();
```

Y se copia el contenido en ellos

```
InputStream in = ConfVM.class.getResourceAsStream("deploy.sh");
OutputStream out = new FileOutputStream(tempDeploy);
FileSystem.copy(in, out);
in.close();
out.flush();
out.close();
```

Por último, lanzamos el script “deploy.sh” que se encarga de ejecutar todo el proceso de configuración de la máquina. Además, una vez que se ha completado todo el proceso, se elimina la carpeta temporal, para que no se almacenen excesivos

archivos y sobrecargar el servidor.

```
Process p = Runtime.getRuntime().exec(
    "/bin/sh " + tempDeploy.getName() + " " + tempSolo.getName()
        + " " + host, null, tempDeploy.getParentFile());

Runtime.getRuntime().exec("rm -rf " + chefDir);
```

5.5.2. Gestión de las máquinas Virtuales

Para la gestión de las máquinas virtuales hemos utilizado la API que proporciona Amazon para este fin.

Nuestro sistema únicamente realiza tres funciones:

- ***“Run”*** para crear y encender una máquina,

```
RunInstancesRequest runInstancesRequest = new RunInstancesRequest(ami, 1, 1);
runInstancesRequest.setInstanceType(type);
runInstancesRequest.setKeyName("");
RunInstancesResult runResult = ec2.runInstances(runInstancesRequest);
Reservation reservation = runResult.getReservation();
List<Instance> instances = reservation.getInstances();
return instances.get(0);
```

- ***“Describe”*** para conocer el estado de una instancia y comprobar que se ha iniciado correctamente

```
DescribeInstancesRequest descreibeInstancesRequest = new DescribeInstancesRequest();
Collection<String> instanceIds = new ArrayList<String>();
instanceIds.add(instanceId);
descreibeInstancesRequest.setInstanceIds(instanceIds);
DescribeInstancesResult result = ec2.describeInstances(descreibeInstancesRequest);
Reservation reservation = result.getReservations().get(0);
List<Instance> instances = reservation.getInstances();
return instances.get(0);
```

- ***"Terminate"*** para apagar la máquina y eliminarla.

```
TerminateInstancesRequest terminate = new TerminateInstancesRequest();
 ArrayList<String> ar = new ArrayList<String>();
 ar.add(idMachine);
 terminate.setInstanceIds(ar);
ec2.terminateInstances(terminate); // visible hasta 1 hora después de eliminar la instancia
```

5.4.3. Generar informes

La generación de informes se realiza a partir de la clase *GenerateReports,* esta clase genera un fichero tipo .xls que contiene un informe en forma de tablas editables, con la información del tiempo de uso de las diversas máquinas del usuario y el coste de estas.

Para llamar a esta clase es necesario pasarle como parámetro el usuario que está logueado. Y gracias a la información del id de usuario se realizan la consultas pertinentes a la Base de Datos para obtener de las máquinas, a partir de sus IDs, la fase del proyecto a la que pertenecen,el tipo de máquina y el tiempo durante el que ha estado lanzada. Todo ello se escribe en un fichero gracias a los recursos que ofrece la biblioteca Apache POI y sobre estos datos se realizan los cálculos del coste de cada máquina individualmente, el coste global para cada fase del proyecto y el coste total del mismo.

Primero creamos un elemento que se encargará de la construcción de nuevos objetos para nosotros y definimos un estilo que usaremos más tarde para ciertas celdas:

```
//Creación del objeto y definicion de estilo
CreationHelper createHelper = wb.getCreationHelper();
sheet = wb.createSheet();
define_Style();
```

Una vez creado el objeto comenzamos a escribir en él los elementos fijos: titulo y fila principal de la tabla:

```
//Titulo de fichero
int k=1;
Row row = sheet.createRow((short) k);
createSet(row,1, "--INFORME--",true);
k=k+2;
//Primera fila de la tabla
row = sheet.createRow((short) k);
createSet(row, 1,"Id maquina",true);
createSet(row, 2,"Tipo",true);
createSet(row, 3,"Tiempo(horas)",true);
createSet(row, 4, "Euros",true);
```

La función createSet inserta el valor de la celda indicada y le asigna estilo si el último parámetro es true. En caso de que el valor que queramos introducir sea una formula lo trataremos de forma diferente:

```
private  void createSet(Row row, int col, String val, Boolean applyStyle){
      Cell cell= row.createCell(col);
      if (val.startsWith("=")){
         cell.setCellFormula(val.substring(1));
      }else{
         cell.setCellValue(val);
      }
      if  (applyStyle)
         cell.setCellStyle(style);
      }
```

El siquiente paso será realizar las consultar para rellenar la tabla:

```
List<Virtual_machines> machines = generalDAO.getMachinesByUser(username);
Iterator<Virtual_machines> it = machines.iterator();
int i=0;
while (it.hasNext()) {
   Virtual_machines vm = it.next();
     row = sheet.createRow((short) k);
     createSet(row,1,vm.getIdMachine(),false);  // idMaquina
     createSet(row,2, vm.getSizeMachine(),false); // tipo maquina
    ....
     createSet(row,3, Long.toString(horas) ,false); //tiempo maquina
```

Y hacer las cuentas necesaria para calcular los costes:

```
createSet(row, 4, "=J"+ ref +"* D"+ (k+1),false ); // Coste maquina
....
```

Por ultimo una vez terminado este proceso se llamará a la función creaFichero que nos creará el elemento .xls, y dará la opción al usuario de guardarlo donde lo estime adecuado (el nombre de este archivo por defecto sera la fecha y hora actual)

```
public File creaFichero() throws IOException, InterruptedException{
    Calendar c = Calendar.getInstance();
    Date hora = c.getTime();
    File infTemp = File.createTempFile("Informe"+hora, ".xls");
    OutputStream out = new FileOutputStream(infTemp);
    wb.write(out);
    out.close();
    return infTemp;
```

6. Conclusiones y trabajos futuros

6.1. Conclusiones

Durante las etapas de diseño, planificación, desarrollo y análisis de Constructor, cada progreso en este proyecto ha supuesto el descubrimiento de ramas del mundo de la informática que el grupo de trabajo no conocía hasta ahora, así como el dominio de lenguajes y herramientas nuevos, la obtención de una visión más completa del estado actual de las tecnologías que han sido empleadas, incluyendo su proyección en un futuro próximo.

Conocimientos adquiridos

La implementación de un sistema de estas características ha supuesto el aprendizaje y perfección de ciertas habilidades que incluyen: la búsqueda selectiva de información e investigación en la Red, artículos y publicaciones, diseño y planificación de proyectos, reparto de tareas y trabajo en equipo, además de técnicas de programación como y nociones de sistemas operativos como:

- Paradigmas de la computación Cloud.
- Dominio de sistemas de ficheros y terminal de comandos en sistemas UNIX/Linux.
- Programación de scripts para la configuración de las máquinas.
- Desarrollo de interfaces gráficas mediante Vaadin.
- Desarrollo de la lógica de la aplicación con Java.

Características y capacidades de constructor

Con este proyecto se han conseguido alcanzar los siguientes objetivos:

- El desarrollo de una aplicación capaz de aprovechar las múltiples cualidades que ofrece la computación Cloud para el almacenamiento y la gestión de datos desde cualquier parte.

- Una aplicación para gestionar, lanzar y eliminar máquinas virtuales de forma cómoda y sencilla.

- Un sistema que permite configurar remotamente máquinas virtuales personalizadas utilizando una interfaz muy intuitiva.

- Desarrollar una herramienta útil para startups y desarrolladores software que ayuda al ahorro en inversión y proporciona un seguimiento del gasto realizado.

Tecnologías desechadas

Al iniciar el desarrollo del proyecto, utilizamos el servidor local Tomcat puesto que era el que más habíamos utilizado los miembros del equipo de desarrollo pero fue sustituido por JBoss 6.1, ya que su uso era más sencillo. Posteriormente tuvimos que cambiar a la versión 4.2.3 ya que había unas librerías en la versión 6.1 que eran incompatibles con el servicio web de AWS.

Como se ha visto en el capítulo 4 de esta memoria, Chef ofrece un amplio abanico de posibilidades para la creación de infraestructuras en máquinas virtuales, en un principio, nos pareció que la opción más adecuada para cumplir nuestros objetivos era Chef-server porque mantenía una estructura cliente-servidor que era lo que nosotros queríamos pero el uso de este tipo de Chef requería tener instalado un Chef-cliente en la máquina remota lo cual hacía necesario que las máquinas virtuales cumplieran una serie de requisitos que reducían el espectro de máquinas virtuales a configurar por lo que nos decantamos por usar Chef-solo que no necesitaba ninguna particularidad en la máquina destino aparte de que su sistema operativo fuera Linux.

En cuanto a la tecnología utilizada para desarrollar la interfaz gráfica, no hubo prueba de ninguna antes de decidirnos por Vaadin, la escogimos principalmente por su sencillez a la hora de desarrollar aplicaciones web, ya que tan sólo hay que preocuparse de la parte del servidor y de usar las componentes visuales, ya que Vaadin gestiona automáticamente la comunicación con la parte cliente. Toda la lógica de la aplicación está en el servidor, lo que hace a la aplicación más segura, más sencilla, más modular y extensible y totalmente completa, ya que se pueden utilizar todas las librerías de Java.

Por último, el servicio de Cloud Computing utilizado ha sido Amazon Web Service, porque creemos que es el servicio más importante, líder en el sector y así podríamos llegar a más gente. Además tiene una excelente relación calidad/precio.

6.2. Trabajos Futuros

Introducción

Dentro del ámbito de un proyecto de estas características, nuestra aplicación web simplemente es un prototipo de lo que podría llegar a ser, pero sirve para ilustrar el funcionamiento de nuestra aplicación y demostrar la potencia de la que puede ser capaz, a pesar de las muchas limitaciones de las que somos conscientes.

A lo largo de todas las etapas de diseño, planificación, desarrollo y análisis del proyecto, se han realizado cambios con vista a mejorar nuestro sistema, ya sea en términos de eficiencia, usabilidad, tiempo, modularidad, etc., pero muchas otras ideas se han quedado en el camino, bien por falta de recursos, de tiempo, o incluso de conocimientos.

Es por estas razones que sería muy provechoso y estimulante enumerar esas ideas de ampliación que el grupo ha planteado en cada etapa del proyecto o aquellas que, a posteriori, una vez finalizado el desarrollo y obtenido las conclusiones, han surgido como si de una tormenta de ideas se tratara. Unas son más factibles para ser desarrolladas por este mismo grupo a partir del sistema descrito en este documento y contando con escasos recursos, otras supondrían contar con un colectivo en el que estuvieran involucrados una cantidad mayor de miembros de proyecto, con grandes conocimientos en muchos ámbitos de la informática y en especial del Cloud Computing.

Diferentes usuarios Amazon

Una de las primeras cosas que se deberían desarrollar en caso de que se quiera dar un uso real a la aplicación es la posibilidad de gestionar distintos usuarios del servicio de Amazon, ya que en el prototipo usamos nuestro usuario por defecto.

El principal motivo por el que no hemos añadido esta opción reside en la complejidad para gestionar el archivo .pem, que es la clave con la que se lanzan las máquinas y con la que se permite la conexión por ssh para enviar los scripts y las recetas a la máquina remota.

Configuración eclipse con MV

Proporcionar a la aplicación la capacidad de configurar eclipse, de tal manera que una vez iniciada la aplicación el usuario fuera capaz de compilar y ejecutar sus proyectos de manera remota en la máquina virtual que estuviera en ese momento lanzada o en la que el usuario especicase. Esto podría permitir que el usuario pudiera estar ejecutando y compilando varios proyectos a la vez, todo ello, incluso, desde una sola máquina física.

Ampliación de Scripts configuración

Poder incluir más recetas en el sistema es algo más ambicioso ya que permitiría ampliar las posibilidades del usuario. Habría varias posibilidades de implementar ésto, una podría ser que únicamente pudiera añadir nuevas recetas el administrador del sistema lo cual sería más fácil de llevar a cabo, ya que el administrador podria meter directamente las recetas en el archivo comprimido que las reúne, y darlas de alta en la base de datos. Aunque para conseguir esto habría que generar un nuevo desplegable de la aplicación.

Otra posibilidad podría ser permitirle al cliente que suba él mismo las recetas, y éstas se darían de alta automáticamente en la base de datos. La complejidad de esta opción reside en cómo están guardadas las recetas, dentro del desplegable, lo que lo hace inviable. Habría que cambiar el modo en el que se guardan las recetas, por ejemplo, almacenarlas en la base de datos.

Posibilidad de añadir nuevas AMIs

Continuando con lo anterior, consistiría en permitir al usuario que pueda añadir las AMIs que desee usar, siempre que sean de Linux, ya que las recetas de Chef que hemos utilizado sólo se pueden usar en Linux. La implementación de esta opción no debe suponer un gran esfuerzo, pues bien se podría poner la tabla editable y recoger los nuevos elementos que se añadan, o bien poner un campo extra para meter los datos de las AMIs, haciendo una comprobación de que los datos son correctos.

Editar los paquetes de Desarrollo y Producción

Al hilo de la edición de las opciones de configuración de la máquina, otra podría ser la posibilidad de permitir al usuario editar los paquetes de “desarrollo” y “producción” que vienen predefinidos, para así permitirle un mayor ahorro de tiempo y una mayor comodidad y personalización.

Almacenar Datos en Amazon

Amazon ofrece multitud de servicios, aunque nosotros únicamente hemos usado Elastic Compute Cloud (EC2). Otro servicio con el que se podría extender la aplicación podría ser utilizando cualquiera dedicado al almacenamiento, para que los datos persistan independientemente de la duración de una instancia.

- Amazon Elastic Block Store (EBS): Diseñados para utilizarlos con las máquinas de EC2, ofrece almacenamiento fuera de la instancia.
- AWS Import/Export: Agilizan la transferencia de grandes cantidades de datos hacia y desde AWS con dispositivos de almacenamiento locales.

Ampliar Servicios de MVs

Una gran ampliación de la aplicación sería la construcción encima de OpenNebula u OpenStack para poder usar clouds híbridos, ya que estos servicios de clouds privados permiten enlazar con servicios públicos (Amazon AWS). De cara al futuro, se puede pensar en la inclusión de otros sistemas de computación cloud que se espera salgan al mercado próximamente, de empresas como Google o Microsoft. Con estas opciones, el usuario tendría la posibilidad de elegir el servicio que fuera más rentable, el más fiable o el que tuviera implementado previamente, y se abriría aún más el campo de uso de la aplicación.

Referencias

- **Amazon**
  - http://aws.amazon.com/es/ec2/

- **Cloud Computing**
  - http://www.channelplanet.com/?idcategoria=23693
  - http://www.societic.com/2010/06/cloud-computing-tipos-de-nubes-de-aplicaciones/
  - http://www.fayerwayer.com/2012/01/el-origen-de-el-computo-en-la-nube/#

- **Chef**
  - http://wiki.opscode.com/display/chef/Home

- **iBATIS**
  - http://ibatis.apache.org/
  - http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iBatis
  - http://blog.mybatis.org/

- **JBoss**
  - http://www.jboss.org/

- **JSON**
  - http://www.json.org/

- **MySQL**
  - http://www.mysql.com/

- **StartUps**
  - http://www.ipvedcor.com/housing.html
  - http://quarkts.com/blog/?p=196
  - http://www.centrodedatos.com/
  - http://www.centrodedatos.com/housing-colocation/
  - http://www.abansys.com/servidores_dedicados_housing.html
  - http://www.unelink.es/housing-30.html
  - http://www.egalan.es/housing.html

- **Vaadin**
  - https://vaadin.com/book

ANEXOS

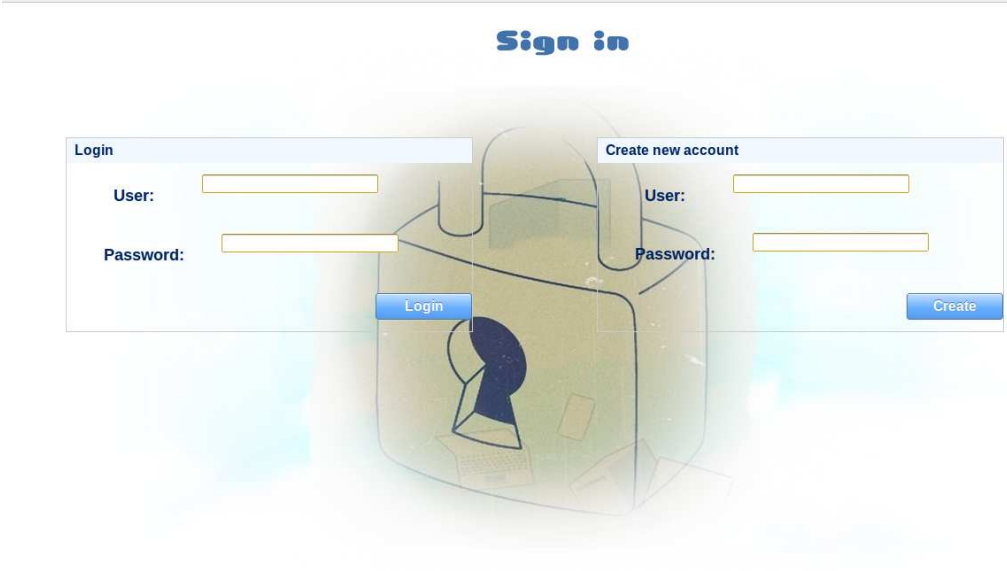
Anexo I: Manual de usuario

El uso de “Constructor” es muy sencillo e intuitivo, dispone de una pantalla inicial para permitir el acceso a los usuarios registrados y a los nuevos usuarios que quieran comenzar a usar “Constructor”; a continuación tenemos una página principal donde cada usuario puede ver las máquinas que tiene lanzadas con diversa información, y una página de configuración donde crear nuevas máquinas y personalizar la configuración de éstas.

En este manual se explican detalladamente cada una de las funciones de la aplicación.

Pantalla “Sign in”

En esta pantalla de inicio, nos encontramos con lo siguiente:



Dispone de dos paneles, el de la izquierda “Login”, para dar acceso a los usuarios que ya se hayan registrado previamente, y el de la derecha “Create new account”, que permite crear nuevos usuarios. Estos nuevos usuarios no pueden estar registrados anteriormente (no se permiten nombres duplicados).

Pantalla principal

Después de loguearnos correctamente, accedemos a la pantalla principal de la aplicación:

Esta pantalla está compuesta por lo siguientes botones:

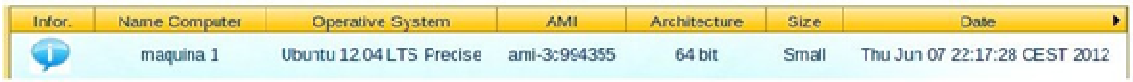
- “New Computer” que nos lleva a la pantalla en la que se pueden crear nuevas máquinas virtuales y gestionarlas.

- “Stop & Remove Computer” que para en Amazon la máquina virtual seleccionada, y la borra del sistema.

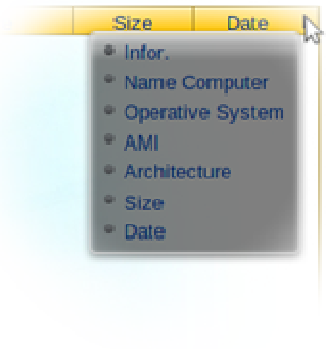
- “Download Report” mediante el cual podemos obtener un resumen del tiempo que ha estado lanzada cada máquina y del coste que ha supuesto.

- “Log Out” para salir de la aplicación.

Y además, en la parte central, podemos ver una tabla, en la que el usuario puede ver sus máquinas virtuales ya creadas y lanzadas y sus características generales, como la fecha en la que fueron lanzadas, el sistema operativo, etc.



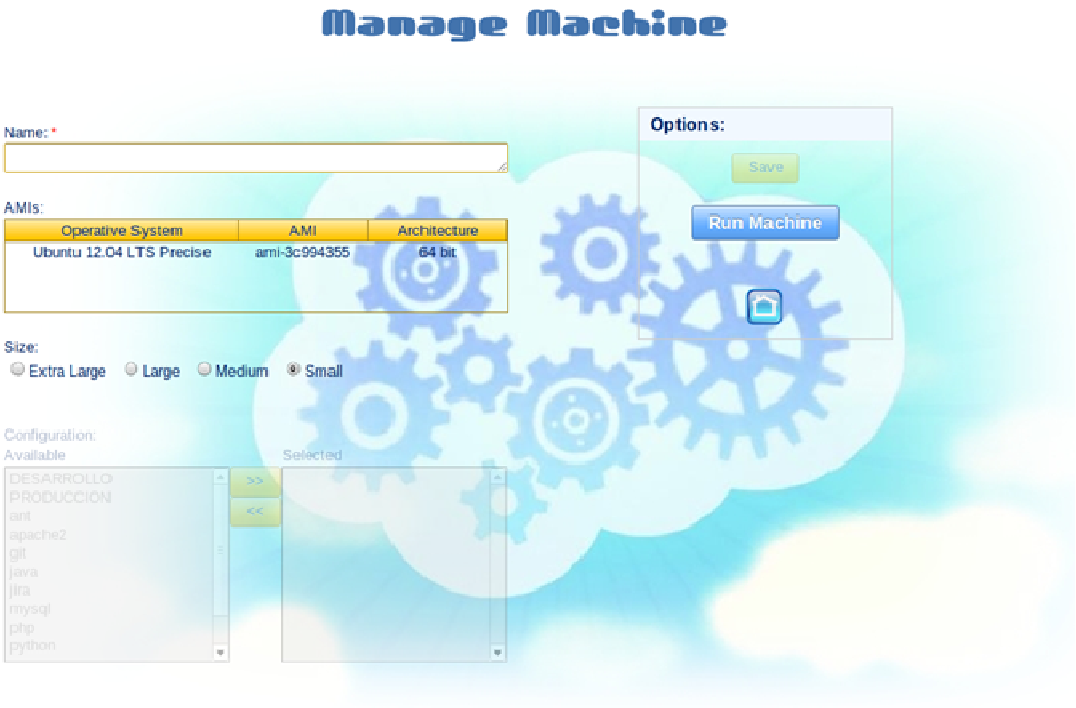
Al pinchar en la flecha de la derecha de la tabla, podemos seleccionar las columnas que deseamos ver.



Dentro del panel informativo, haciendo “click” sobre el icono azul de ‘Info’ aparece una nueva ventana, en la que se especifican los paquetes que tiene instalada la máquina; pulsar en el nombre de la máquina nos permite acceder a la pantalla en la que podemos ver más detalladamente diversa información de la máquina, así como modificar algunos parámetros.

Pantalla “Manage Machine”

A esta pantalla se accede accionando el botón “New Computer” de la pantalla principal.



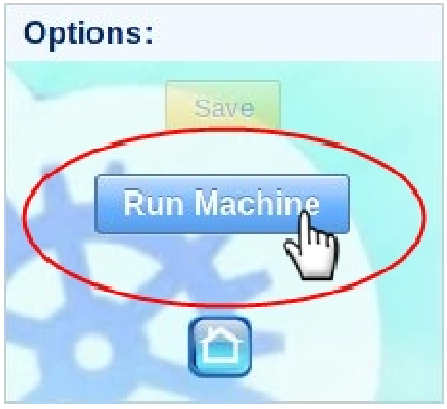
Para crear una nueva máquina debemos seguir los siguientes pasos:

1. Insertar un nombre en el campo “Name”
2. Seleccionar una imagen de máquina de Amazon (AMI),que es simplemente un entorno empaquetado que incluye todos los bits necesarios para configurar y arrancar la instancia. Esta imagen debe contener el sistema operativo Linux.

3. Elegir un tamaño:

   - Small: 1,7 GB de memoria, 1 unidad de sistemas EC2 (1 núcleo virtual con 1 unidad de sistemas EC2), 160 GB de almacenamiento de almacenamiento de instancia local, plataforma de 32 o 64 bits
   - Medium: 3,75 GB de memoria, 2 unidades de sistemas EC2 (1 núcleo virtual con 2 unidades de sistemas EC2 cada uno), 410 GB de almacenamiento de instancias local, plataforma de 32 o 64 bits.
   - Large: 7,5 GB de memoria, 4 unidades de sistemas EC2 (2 núcleos virtuales con 2 unidades de sistemas EC2 cada uno), 850 GB de almacenamiento de instancias local, plataforma de 64 bits.
   - Extra Large: 15 GB de memoria, 8 unidades de sistemas EC2 (4 núcleos virtuales con 2 unidades de sistemas EC2 cada uno), 1690 GB de almacenamiento de instancias local, plataforma de 64 bits.

4. Hacer “click” sobre el botón “Run Machine”.



Mientras se están lanzando las máquinas se deshabilitan los botones hasta que se completa el arranque de la máquina.

Los precios de las máquinas de Amazon en función del tamaño son los siguientes:

Tamaño	Uso de Linux/UNIX
Pequeño	$0,080 por hora
Mediano	$0,160 por hora
Grande	$0,320 por hora
Extragrande	$0,640 por hora

Configuración de las máquinas

Después de esto se puede proceder a la configuración de la máquina eligiendo las características que queremos que contenga la máquina, ya sea un gestor de bases de datos o una versión específica de un lenguaje de programación. Para facilitar esta configuración, el usuario, dispone de dos “paquetes” predeterminados que contienen las características más comunes que tiene que tener un ordenador para las fases de desarrollo y producción de un producto software.

La configuración elegida se guarda pulsando en el botón “Save”, aunque esto sólo guarda los cambios en nuestro sistema, útil para parámetros como el nombre.

Para configurar la máquina virtual de Amazon, debemos pulsar el botón “Configure Machine”,

Que ejecutará los archivos de configuración sobre la máquina en que estamos trabajando, y nos mostrará un mensaje cuando haya finalizado la configuración y ya podamos utilizarla.

El panel de información, muestra el tiempo que lleva lanzada la máquina, el coste que esto supone dependiendo del tamaño de la máquina, y la DNS (sistema

de nombres de dominio) de la máquina virtual. La DNS es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.

Al lanzar una máquina de AWS nos proporciona una DNS única para poder conectarnos a la máquina por ejemplo mediante ssh, con la clave única del usuario, y el nombre de host: ubuntu@ec2-23-20-247-64.compute-1.amazonaws.com

Si pulsamos en el botón Home,

Volvemos a la pantalla principal.

Descargar Informes

Si pulsamos en el botón “Download Report” en el Menú Principal,

Se genera un informe con formato ‘.xls’ que contiene la información de las máquinas virtuales que hay arrancadas y el coste de cada máquina en función del tamaño de éstas.

Dependiendo del navegador, nos da la posibilidad de guardar el archivo en la ubicación que deseemos del ordenador, o nos descarga el archivo directamente en nuestra carpeta predefinida de descargas.

Requisitos

La aplicación web es compatible con todas las plataformas (Windows, Linux, IOS…) y todos los navegadores actuales, únicamente es necesario una conexión a internet.

Anexo II: Eclipse Remote Control

Es un plug-in para eclipse que proporciona funciones de control remoto para eclipse.
Los comandos pueden ser enviado via el eclipse de control remoto a una instancia de eclipse en ejecución.

El sitio de actualizaciones de Eclipse está disponible aquí: https://github.com/marook/eclipse-remote-control/raw/master/workspaces/erc/update_site/site.xml

Construcción

Para construir el cliente de control remoto de eclipse hay que usar el fichero ant build.xml en el proyecto cliente. Los siguiente
To build the eclipse remote control client use the ant build.xml file. Los siguientes objetivos ant estan disponibles:

- Clean: Elimina todos los recursos temporales necesarios para la construcción de ant.
- Compile: Compila el código para el control cliente remoto de eclipse.
- Package: Creates a jar file with the client application. Crea un fichero jar con la aplicación cliente.

Utilice el paquete de destino por defecto para crear la aplicación cliente.

Uso

Los requisitos son:

- Se han instalado los ficheros de control remoto de plugins de Eclipse.
- Usted ha descargado el paquete de cliente de eclipse de control remoto.

Inicie el plugin eclipse de control remoto instalado.
Después de que Eclipse haya comenzado usted puede enviar comandos a través del control remoto de la aplicación cliente.

La sintaxis es:

$ Java-jar eclipse_remote_control.jar [comando] [argumentos de comandos]

Comandos

En la actualidad los comandos siguientes son compatibles:

- **Open File**

Este comando abre un archivo en la instancia de Eclipse en ejecución.

El archivo se busca primero en el área de trabajo de Eclipse. Si el archivo no se puede encontrar en el espacio de trabajo Eclipse, entonces se abre el archivo de sistema de archivos del sistema operativo.

Un número de línea opcional se puede especificar los archivos que se abren desde el espacio de trabajo de Eclipse. Después de abrir el archivo especificado, la línea será seleccionada.

Sintaxis:

```
$ java -jar eclipse_remote_control.jar open_file [absolute path to file] <line number>
```

Ejemplo:

```
$ java -jar eclipse_remote_control.jar open_file /tmp/file.txt 2
```

- **Execute Command**

Este comando se utiliza para ejecutar configuraciones de ejecución o herramientas externas de configuración. Esto incluye ‘ant builds’ por ejemplo.

Usted tiene que especificar el nombre de la configuración que se esta ejecutando. Para saber el nombre de la configuración que se esta ejecutando -> Ejecutar configuraciones... en eclipse.

Seleccione la configuración de ejecución en el árbol de la izquierda. Entonces el nombre se muestra en la primera fila en el lado derecho.

Como un argumento opcional, puede pasar el modo de ejecución en el que se puso en marcha la configuración de ejecución. Las alternativas posibles son los siguientes:

- RUN
- DEBUG
- PERFIL

Sintaxis:

$ java -jar eclipse_remote_control.jar execute_command [run configuration name] <run mode>

Ejemplo:

$ java -jar eclipse_remote_control.jar execute_command my_ant_config DEBUG